

SHARP®

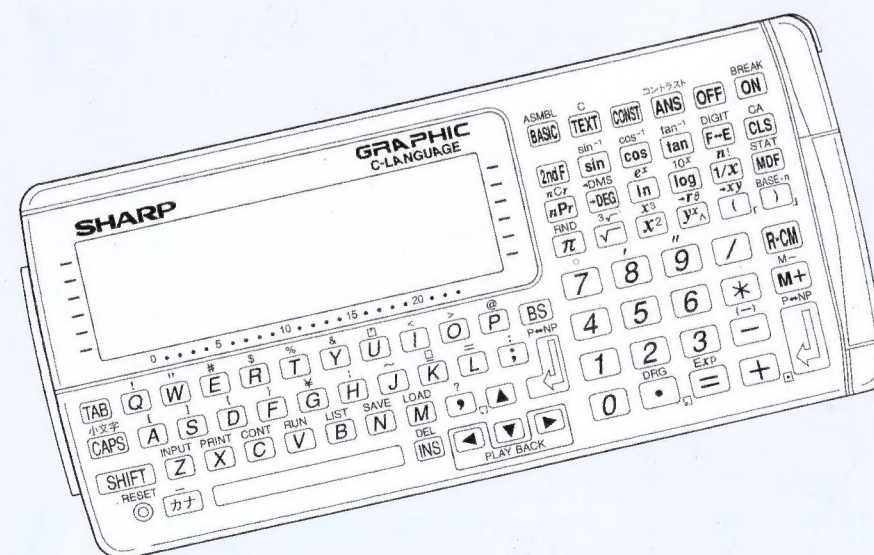
取扱説明書

ポケットコンピュータ

形名 **PC-G850 VS**

保証書付 (巻末)
(WITH WARRANTY CARD)

	ページ
●お使いになる前に	12
●基本操作とモードについて	17
●第1章 マニュアル計算	25
●第2章 関数計算の操作方法と練習	38
●第3章 算術代入計算	70
●第4章 BASIC言語	97
●第5章 TEXTモード	169
●第6章 C言語機能	185
●第7章 CASL	244
●第8章 機械語モニタとアセンブラ機能	273
●第9章 PIC	306
●第10章 BASICの各命令の説明	315
●付録	371



安全にお使いいただくために




図記号について

この取扱説明書には、安全にお使いいただくためにいろいろな表示をしています。その表示を無視して誤った取り扱いをすることによって生じる内容を、次のように区分しています。

内容をよく理解してから本文をお読みになり、記載事項をお守りください。

- ⚠ 警告** 人が死亡または重傷を負うおそれがある内容を示しています。
- ⚠ 注意** 人がけがをしたり財産に損害を受けるおそれがある内容を示しています。

図記号の意味

-  記号は、気をつける必要があることを表しています。
-  記号は、してはいけないことを表しています。
-  記号は、しなければならないことを表しています。

⚠ 警告

- この製品およびACアダプターの上やそばに水やコーヒーなど液体の入った容器を置かないでください。倒れて内部に水などが入ると、火災や感電の原因となります。
- お客様による改造や修理はしないでください。火災や感電、けがの原因となります。
- この製品には、EA-23E以外のACアダプターは使用しないでください。他のACアダプターをご使用になると、故障や火災の原因となります。
- ACアダプターの電源はAC100V（50/60Hz）のコンセントを使用してください。それ以外の電源で使用されますと、火災の原因となります。
- ACアダプターをコンセントに直接接続してください。タコ足配線は加熱し、火災の原因となります。
- ぬれた手でACアダプターを抜き差ししないでください。感電の原因となります。
- 電源コードを傷つけたり、破損したり、加工したりしないでください。また重いものを載せたり、引っ張ったり、無理に曲げたりすると電源コードを傷め、火災や感電の原因となります。
- 万一、発熱していたり、煙が出ている、変な臭いがするなどの異常状態のままで使用しますと、火災、感電の原因となります。すぐにこの製品の電源スイッチを切り、ACアダプターをコンセントから抜き、お買いあがの販売店にご連絡ください。
- 雷がなりはじめたら、落雷による感電・火災の防止のため、この製品の電源を切り、ACアダプターをコンセントから抜いてください。



⚠ 注意

- 使用しないときはACアダプターをコンセントおよびACアダプター接続端子から外しておいてください。
- この製品およびACアダプターは、ほこりや湿気の多い場所で使用しないでください。ほこりや汚れ・水滴がつきますと、火災や感電・漏電の原因となることがあります。
- 電池は誤った使いかたをすると、破れつや発火の原因となることがあります。また、液もれして機器を腐食させたり、手や衣服などを汚す原因となることもあります。以下のことをお守りください。
 - ・プラス⊕とマイナス⊖の向きを表示どおり正しく入れる。
 - ・種類の違うものや新しいものと古いものを混ぜて使用しない。
 - ・使えなくなった電池を機器の中に放置しない。
 - ・端子をショートさせない。
 - ・水や火の中に入れたり、分解しない。
 - ・もれた液が体についたときは、水でよく洗い流す。
 - ・充電電池（ニカド電池）は使用しない。
- ACアダプターを抜くときは、電源コードを引っ張らないでください。コードが傷つき、火災、感電の原因となることがあります。



<はじめに>

お買いあげいただき、まことにありがとうございました。

この取扱説明書をよくお読みのうえ、正しくお使いください。

ご使用前に、「安全にお使いいただくために」を必ずお読みください。

この取扱説明書は、いつでも見ることができる場所に必ず保存してください。

本書で使用する用語は、情報処理技術者用の一般書籍にできるだけ準じています。

<ご使用前のおことわり>

- お客様または第三者がこの製品および付属品の使用を誤ったことにより生じた故障、不具合、またはそれに基づく損害については、法令上の賠償責任が認められる場合を除き、当社は一切その責任を負いませんので、あらかじめご了承ください。
- この製品は付属品を含め、改良のため予告なく変更することがあります。

<おことわり>

本書に記載の別売周辺機器（CE-126P（プリンタ）、CE-T800（パソコン通信ケーブル）、EA-129C（ポケコン接続ケーブル））は2007年12月現在、販売を完了しております。

あらかじめご了承ください。

すでに上記周辺機器をお持ちの場合は、この製品に接続してお使いいただけます。

<記憶内容保存のお願い>

故障・修理のときや電池交換を行ったときは、記憶内容が消失します。また、この製品の使いかたを誤ったときや静電気・電氣的ノイズの影響を受けたときは、記憶内容が変化・消失する恐れがあります。
重要な内容は必ず紙などに控えておいてください。

この取扱説明書の内容にあたっては、新潟県下の工業高校（新潟県立長岡工業高等学校、新潟県立三条工業高等学校、新潟県立新潟工業高等学校、新潟県立巻工業高等学校、新潟県立高田工業高等学校、新潟県立小千谷西高等学校）ならびに岐阜県立岐阜工業高等学校の先生方、および関係者の方々に多大なご協力をいただきました。

この場をお借りし、心から感謝申し上げます。

（平成13年7月現在）

◆お使いになる前に	(ページ) 12
1. おねがい	12
2. お買いあげ後はじめてご使用になるときの操作	14
3. 各部のなまえ	16
◆基本操作とモードについて	17
1. 電源のオン／オフ(入／切)と表示の濃度調整	17
2. モードについて	18
3. 基本的なキー操作	19
4. 表示シンボルについて	21
5. カタカナの入力のしかた	23
第1章 マニュアル計算(手操作による計算)	25
1. マニュアル計算	25
2. マニュアル計算のしかた	25
3. キー操作の練習(訂正のしかた)	26
4. エラーの処理について	29
5. 数式が長い場合の連続計算のしかた	30
6. ラストアンサー機能について	30
7. 計算結果の表示方法	31
8. 表示に関するフォーマット指定	32
9. 数値丸め機能 MDF(モディファイ)	34
10. 計算結果の符号の反転	35
11. メモリ計算	35
12. 定数計算機能	36
第2章 関数計算の操作方法と練習	38
1. マニュアル(手操作)での関数計算	38
2. マニュアル計算における関数計算の操作方法と練習	45
① 2乗	45
② 平方根	46
③ 3乗	47
④ 立方根	47
⑤ 逆数	48
⑥ べき乗	48
⑦ べき乗根	49
⑧ 階乗・順列・組み合わせ	49
⑨ 常用対数	50
⑩ 常用指数	51
⑪ 自然対数	52

⑫ 自然指数	53
⑬ 三角関数	54
⑭ 逆三角関数	58
⑮ 座標変換	59
⑯ 統計計算	62

第3章 算術代入計算

1. 算術代入計算	70
2. 例題と解説	71
・マニュアル計算の練習問題	74

第4章 BASIC言語

1. BASIC言語をマスターする第一歩	97
2. プログラムの基本	102
STEP① INPUT、PRINT、END、GOTO文	102
STEP② 切り捨て・四捨五入・桁指定	110
STEP③ 関数を使うプログラム	113
STEP④ IF～THEN～ELSE	117
STEP⑤ FOR～TO～STEP、NEXT	121
STEP⑥ REM、READ、DATA、RESTORE	125
STEP⑦ GOSUB～RETURN	128
STEP⑧ 配列 DIM(ディメンジョン)	129
STEP⑨ USING(ユージング)、PRINT USING	137
STEP⑩ MIDS(ミッド・ドル)、LEN(レングス)、VAL(バリュー)	138
STEP⑪ CHR\$(キャラクタドル)、STR\$(ストリングドル)	139
STEP⑫ ASC(アスキー)、論理演算子	140
3. 構造化BASIC命令の使いかた	143
4. スクリーンエディタについて	145
・BASICによるプログラム演習問題	148
5. 変数の種類と使いかた	153
6. デバッグ	156
7. プログラムのファイル	158
8. データのファイル	160
9. 別のポケコンへの記録、読み込み	161
10. プログラムの実行開始方法とラベルについて	162
・n進演算機能	164

第5章 TEXTモード(テキストエディタ)

1. TEXTモード機能一覧	170
2. TEXTモードの使いかた	171
2.1 TEXTモードの設定	171
2.2 エディット機能(Edit)	171

2.3	TEXTプログラムの消去>Delete)	174
2.4	TEXTプログラムの印字(Print)	174
2.5	シリアル入出力(Sio)	174
2.6	ミニI/OからのTEXTプログラム送出	177
2.7	プログラムファイル(File)	177
2.8	BASICコンバータ(Basic)	179
2.9	ラムデータファイル(Rfile)	180

第6章 C言語機能185

1.	C言語の特徴	186
2.	始めようCプログラミング	187
2.1	Cプログラムの作成から実行まで	187
2.2	プログラムのトレース実行	190
2.3	表示出力と印字出力の切り替え	191
2.4	C言語機能一覧	192
2.5	Cプログラムのスタイル	193
3.	基本的なCプログラミング	195
3.1	整数を扱うプログラム	195
3.2	実数を扱うプログラム	198
4.	プログラムの流れを制御する	199
4.1	条件文の使いかた	199
4.2	繰り返し文の使いかた	201
5.	C言語らしいプログラミング	204
5.1	配列	204
5.2	関数	207
5.3	ポインタ	211
6.	C言語の要点	213
6.1	本機のC言語仕様	213
6.2	キーワード	214
6.3	定数	215
6.4	扱える数値の範囲	216
6.5	式と文	216
6.6	演算子	216
6.7	いろいろな構文	220
6.8	記憶クラス	223
6.9	多次元配列	223
6.10	構造体	224
6.11	プリプロセッサ	224
7.	ライブラリ関数	226
7.1	標準入出力関数	226
7.2	文字処理関数	231
7.3	文字列処理関数	232

7.4	メモリ割り当て関数	233
7.5	数学関数	233
7.6	ハードウェア・インタフェース関数	234
7.7	データファイル関数	237
7.8	グラフィック関数	238
7.9	その他の関数	239
8.	エラーメッセージ	241
8.1	コンパイルエラーメッセージ	241
8.2	実行時エラーメッセージ	243

第7章 CASL244

1.	CASL(アセンブラ言語)	244
2.	CASLモードの構成	244
2.1	プログラム作成手順	245
3.	ソースプログラムの作成、編集(エディット)	247
3.1	ソースプログラムの入力形式	247
3.2	ソースプログラムの消去	248
3.3	ソースプログラムの入力	248
4.	アSEMBル	250
4.1	アSEMBルリストをプリンタで印字する方法	250
4.2	エラーメッセージ	251
5.	シミュレーション	251
5.1	ノーマル実行	252
5.2	トレース実行	252
5.3	シミュレーションでのエラー	253
6.	モニタ	254
6.1	レジスタの内容の表示	254
6.2	レジスタに値を設定する方法	255
6.3	オブジェクトコードの表示	256
6.4	任意のアドレス内容の表示	257
6.5	オブジェクトコードの書き換え	257
7.	CASL実行例	257
7.1	オブジェクトコードの内容確認	258
7.2	ノーマル実行	259
7.3	ブレークポイントの解除	261
7.4	トレース実行	261
7.5	空欄穴埋め問題の実行例	263
8.	COMETの仕様	265
8.1	仮想計算機COMETと本機CASLとの相異点	265
8.2	COMETの仕様概略	266
8.3	命令語の構成	266
8.4	命令の種類と機能	267

8.5 アセンブラの文法	(ページ) 270
8.6 疑似命令	270
8.7 マクロ命令	271
8.8 特殊命令	272
第8章 機械語モニタとアセンブラ機能	273
機械語モニタ機能	274
1. 機械語モニタを使ううえでのきまり	274
2. 機械語モニタの各命令の説明	275
* USER ユーザーエリア	275
* S セットメモリ	276
* D ダンプメモリ	277
* E イクスチェンジメモリ	278
* P プリントスイッチ	279
* G ゴーサブ	279
* R リードSIO	280
* W ライトSIO	280
* BP ブレークポイント	281
3. 機械語モニタモードでのエラー表示	283
アセンブラ機能	284
1. ソースプログラムと機械語プログラム	284
2. アセンブラの機能	285
3. アセンブルしてみましょう	288
4. ソースプログラムの作成・編集	291
4.1 ソースプログラムの構成	291
4.2 ソースプログラムの消去	294
4.3 ソースプログラムの入力	294
5. アセンブル	294
5.1 メニュー画面	294
5.2 アセンブルの実行	296
5.3 アセンブルリストを表示させる方法	297
5.4 アセンブルリストを印字させる方法	298
・アセンブルリストをミニI/Oから送出する方法	299
6. アセンブラの疑似命令の説明	300
ORG	300
DEFB/DB/DEFM/DM	301
DEFW/DW	303
DEFS/DS	303
EQU	304
END	304
7. アセンブラでのエラー	304

第9章 PIC	(ページ) 306
1. PIC	306
2. PICプログラムの作成手順	306
2.1 プログラム作成手順	307
2.2 機械語エリア(ユーザーエリア)の確保	308
3. ソースプログラムの作成・編集	308
3.1 ソースプログラムの構成	308
3.2 ソースプログラムの消去	310
3.3 ソースプログラムの入力	310
4. アセンブラ	311
4.1 アセンブラの疑似命令	311
4.2 #INCLUDE	312
4.3 エラーメッセージ	313
第10章 BASICの各命令の説明	315
AND (論理積、条件式の結合)	316
ASC (文字などをキャラクタコードに変換)	316
AUTO (行番号の自動発生)	317
BLOAD (別のポケコンからBASICプログラムを読み込む)	317
BLOAD M (別のポケコンから機械語プログラムを読み込む)	318
BLOAD ? (BASICプログラムを照合)	318
BSAVE (BASICプログラムを別のポケコンに記録)	319
BSAVE M (機械語プログラムを別のポケコンに記録)	319
CALL (機械語プログラムを実行)	319
CHR\$ (キャラクタコードを文字や記号(キャラクタ)に変換)	320
CIRCLE (画面に円を描く)	320
CLEAR (変数を消去)	322
CLOSE (入出力のファイルを閉じる)	322
CLS (表示内容を消去)	323
CONT (中断したプログラムを再実行)	323
DATA (変数に与えるデータを指定)	323
DEGREE (角度単位を“度”に設定)	323
DELETE (プログラムの行を削除)	323
DIM (配列変数を確保)	324
DMS\$ (10進数の数値を60進数の文字列に変換)	325
END (プログラムの実行を終了)	326
EOF (ファイルの終わりを検出)	326
ERASE (配列変数を消去)	326
FILES (プログラムファイルエリアのファイル名とファイルサイズを表示)	327
FIX (数値の整数部を求める)	327

FOR~NEXT	(繰り返し命令)	327
FRE	(プログラム・データエリアの空き容量を求める)	328
GCURSOR	(グラフィック表示の位置指定)	329
GOSUB~RETURN	(サブルーチンジャンプと復帰命令)	329
GOTO	(ジャンプ命令)	330
GPRINT	(グラフィック表示命令)	330
GRAD	(角度単位をグラードに設定)	332
HEX\$	(16進数文字列への変換)	333
IF~THEN~ELSE	(条件判断命令)	333
IF~THEN~ELSE~ENDIF	(条件判断命令)	334
INKEY\$	(押されたキーの内容を読み込む)	335
INPUT	(データの入力命令)	336
INPUT#	(ファイルのデータを読み込む)	337
KILL	(プログラムファイルエリアのファイルを消去)	338
LCOPY	(プログラム行を複写)	338
LEFT\$	(文字列の左側から指定した文字数分を取り出す)	339
LEN	(文字列の文字数を求める)	339
LET	(変数に数値や文字を代入)	339
LFILES	(プログラムファイルエリアのファイル名を印字)	339
LINE	(表示上の2点間を線で結ぶ)	340
LIST	(プログラムを表示)	341
LLIST	(プログラムを印字)	341
LNINPUT#	(ファイルの1行単位のデータを読み込む)	342
LOAD	(プログラムファイルエリアのファイルを読み出す)	342
LOCATE	(表示の開始位置を指定)	343
LOF	(ラムデータファイルの未使用領域の大きさを求める)	343
LPRINT	(指定した内容を印字)	344
MID\$	(文字列の中から指定した文字数分を取り出す)	344
MON	(機械語モニタに入る)	345
NEW	(プログラム、データを消去)	345
NOT	(与えられた数値の否定を取る)	345
ON~GOTO	(指定された行を選択して実行を移す)	346
ON~GOSUB		
OPEN	(入出力の回路を開く)	346
OR	(論理和、条件式の結合)	347
PAINT	(指定したパターンで領域を塗りつぶす)	347
PASS	(パスワードの設定、解除)	348
PEEK	(機械語プログラムやデータを読み出す)	349
POINT	(表示ドットの状態を読み取る)	349
POKE	(機械語プログラムやデータを書き込む)	350
PRESET	(表示ドットを消去)	350
PRINT	(表示命令)	351

PRINT#	(データの記録、送出命令)	352
PSET	(表示ドットを点灯または反転)	354
RADIAN	(角度単位をラディアンに設定)	354
RANDOMIZE	(乱数の種を植えつける)	354
READ	(データを変数に読み込む)	355
REM	(プログラムに注釈を入れる)	355
RENUM	(プログラムの行番号をつけ直す)	356
REPEAT~UNTIL	(繰り返し命令)	356
RESTORE	(READ文で変数に入れるデータの順番を変える)	357
RIGHT\$	(文字列の右側から指定した文字数分を取り出す)	358
RND	(乱数(疑似乱数)を発生させる)	358
RUN	(プログラムの実行を開始)	359
SAVE	(BASICプログラムをプログラムファイルエリアに登録)	359
STOP	(プログラムの実行を一時停止)	359
STR\$	(数値を文字列に変換)	360
SWITCH~CASE~DEFAULT~ENDSWITCH	(分岐型のジャンプ)	360
TROFF	(トレースモードを解除)	361
TRON	(トレースモードを設定)	361
USING	(表示、印字のフォーマット指定)	362
VAL	(文字列を数値に変換)	363
VDEG	(60進数の文字列を10進数の数値に変換)	363
WAIT	(PRINT命令の停止時間を指定)	364
WHILE~WEND	(繰り返し命令)	364
XOR	(排他的論理和、条件式の結合)	365
ミニ1/Oに関する命令		
INP	(入力ポートからの入力関数)	366
OUT	(出力ポートへの出力命令)	367
OPEN	(出力デバイスを指定)	367
CLOSE	(デバイス指定を解除)	368
LLIST	(プログラムをパラレルポートから出力)	368
LPRINT	(指定した内容をパラレルポートから出力)	368
8ビット制御に関する命令		
PIOSET	(入出力モードの設定)	369
PIOGET	(入力ポートからの入力関数)	369
PIOPUT	(出力ポートへの出力命令)	369
システムバス入出力命令		
INP	(指定したポートからの入力関数)	370
OUT	(指定したポートへの出力命令)	370

付 録371

1. パソコン通信ケーブルCE-T800について371
2. 電池の交換について372

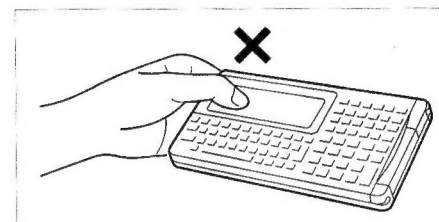
3. 主なキーの主な機能	(ページ) 375
4. 計算範囲	377
5. 仕様	379
● 異常が発生した場合の処理	380
● システムバス端子信号表	381
● ローマ字→カナ変換表	382
● キャラクタ・コード表	384
● メモリマップ・I/Oマップ	385
● エラーコード表	386
● 故障かな?と思ったら	388
● アフターサービスについて	389
● 保証書	392

本機と周辺機器（プリンタなど）接続時のご注意

- 本機の電源を切ってから、周辺機器と接続してください。
電源を入れたままで接続すると、故障の原因となります。

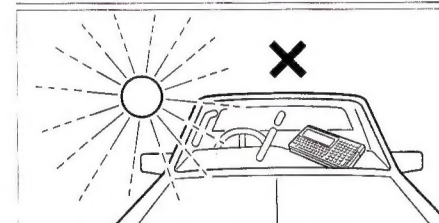
お使いになる前に

1. おねがい

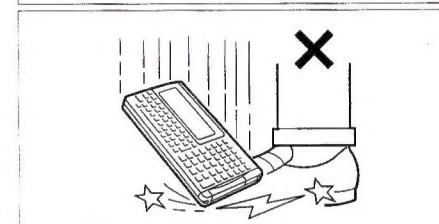


液晶表示部を強く押さえないでください。

表示部はガラスでできていますので、押さえると割れることがあります。



日の当たる自動車内・直射日光が当たる場所・暖房器具の近くなどに置かないでください。
高温により、変形や故障の原因になります。



落としたり、ぶつけたり、強いショックを与えないでください。
大きな力加わり、壊れることがあります。



お手入れは、乾いたやわらかい布で軽くふいてください。シンナーやベンジンなど、揮発性の液体やぬれた布は使用しないでください。変質したり色が変わったりすることがあります。

本体裏面に貼っているネームラベルにお名前をご記入ください

- カバンには硬いものや先のとがった物と一緒に入れないでください。傷がつくことがあります。また、ハードカバーを必ず取り付けてください。
- 防水構造になっていませんので、水など液体がかかる場所での使用や保存は避けてください。雨、水しぶき、ジュース、コーヒー、蒸気、汗なども故障の原因となります。
- 本機に周辺機器（プリンタなど）を接続するときは、本機の電源を切ってから行ってください。電源を入れたままで接続すると、故障の原因となります。

この装置は、情報処理装置等電波障害自主規制協議会（VCCI）の基準に基づくクラスB情報技術装置です。この装置は、家庭環境で使用することを目的としています。この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。

取扱説明書に従って正しい取り扱いをしてください。

正しい取り扱いをしても、電波の状況によりラジオ、テレビジョン受信機の受信に影響を及ぼすことがあります。そのようなときは、次の点にご注意ください。

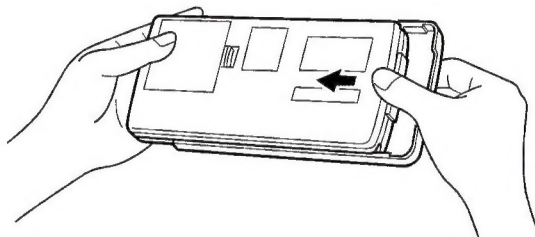
- この製品をラジオ、テレビジョン受信機から十分に離してください。
- ACアダプターとラジオ、テレビジョン受信機を別のコンセントに接続してください。
- 使用されるケーブルは指定のものを使用してください。

ハードカバーの使いかた

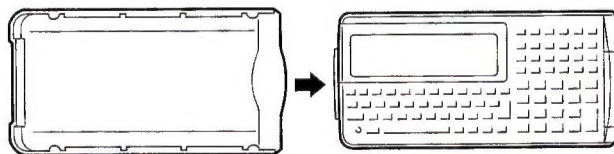
ハードカバーは衝撃に対して本体を保護する役割を持っています。

ポケットコンピューターを使用しないとき、また、カバンに入れて持ち運ぶときなどにはハードカバーを取り付けてください。

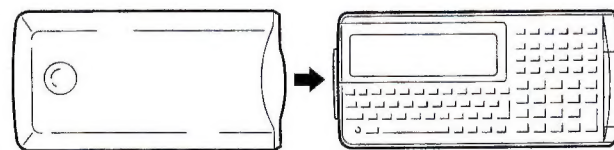
ハードカバーの取り外しかた



使うとき



使わないとき

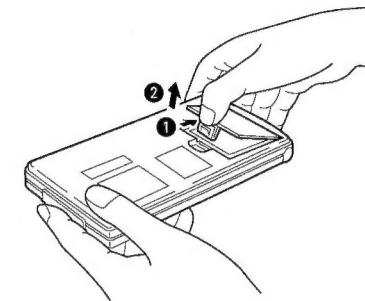


2. お買いあげ後はじめてご使用になるときの操作

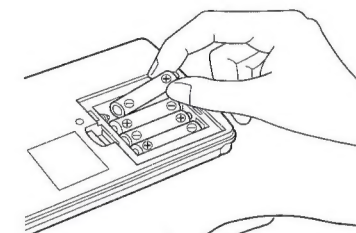
(1) 電池の取り付け

付属の乾電池を次の手順で入れてください。

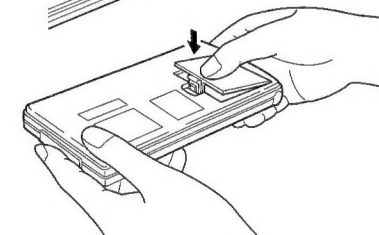
① 本体裏面の電池ぶたを図のようにして外します。



② 乾電池をマイナス⊖側から入れます。⊕・⊖をまちがえないように入れてください。



③ 電池ぶたをもとどおり取り付けます。

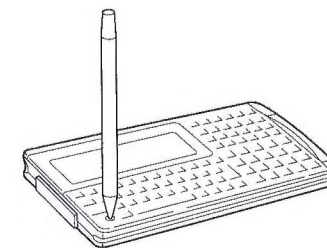


(2) 初期設定

電池を入れた直後は、本機の内部が不安定な状態になっています。次の操作で初期設定を行ってください。初期設定をしないでいると、電池が早く消耗します。

① **[ON]** キーを押して電源を入れた後、ボールペンなどで、本体左端のリセット（RESET）スイッチを押してください。

芯先の出たシャープペンシルや先の折れやすいもの、また、針など先のとがったものは使用しないでください。




- ②リセットスイッチを離すと次の画面になります。違う画面になったときはもう一度リセットスイッチを押してください。


MEMORY CLEAR O. K. ? (Y/N)

(メモリー内容を消去しますか?)

本書の表示例では、説明上必要なシンボルだけを記載しています。(シンボルについては22ページを参照してください。)

- ③  キーを押してください。次の画面(点滅)になります。
(初期設定し、記憶内容をすべて消去したことを示しています。)



ALL RESET

- ④  キーを押してください。次の画面になります。

RUN MODE
>



キーの表記について

説明中の  や  は、それぞれのキーを押すことを示しています。

最初は説明どおりにキー操作を行い、ポケットコンピュータになれましょう。

(3) 動作確認

正常に動作しているか確認するために、次のキー操作を行ってください。




RUN MODE
FRE 30179.

以上のように表示すれば本機は正常に動作しており、指令待ち状態になっています。

最後に表示された“30179.”という値は、本機にプログラムやデータを入れることができる最大の容量を示しています。

(注) (2)~(3)の操作を行っても説明どおりにならない場合は、もう一度読み直して、操作をやり直してください。

電池交換について

“ ”シンボルが画面左下に点灯したときは、本機内の電池が消耗したことを示しています。このシンボルが点灯したときは、一度  で電源を切り、 で再度電源を入れてください。それでもなおこのシンボルが消えないときは、速やかに372ページの手順に従って電池を交換してください。このシンボルが点灯したままですと、突然電源が切れたり、電源が入らなくなったりします。

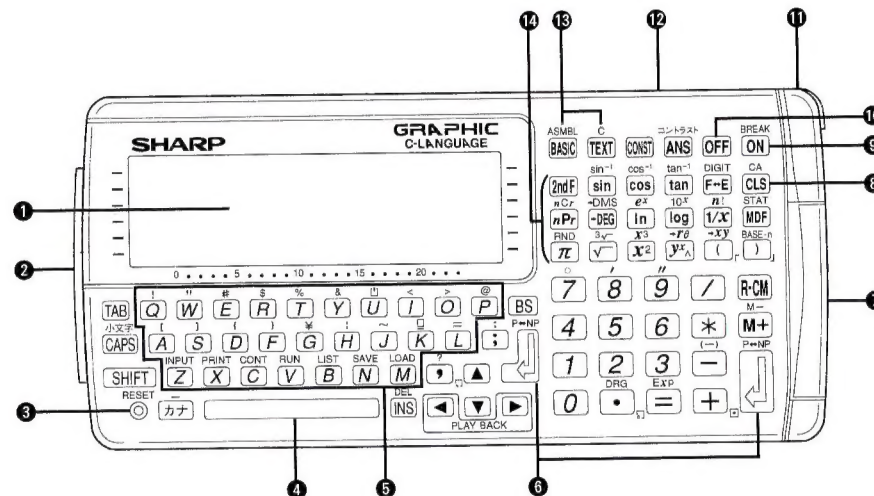
(注)・電池を交換するとプログラムやデータが消えます。

電池を交換する前にプログラムやデータを紙に書き写しておいてください。周辺機器 C E -126 P (プリンタ) をお持ちの場合は印字しておいてください。



- 別売の A C アダプター E A -23 E をご使用の場合も、A C アダプターを抜いて、本機内の電池が消耗していないか確かめてください。

電池が消耗していると、A C アダプターを取り外したときにプログラムやデータが消えてしまいます。

3. 各部のなまえ



- ① 表示部
- ② 周辺機器接続端子 (11ピン)
- ③ リセットスイッチ
- ④ スペースキー
- ⑤ アルファベットキー
- ⑥ キャリッジリターンキー
- ⑦ システムバス端子
- ⑧ クリア/クリアオールキー
- ⑨ 電源オン/ブランクキー
- ⑩ 電源オフキー
- ⑪ A C アダプター接続端子
- ⑫ 電池ふた (裏面)
- ⑬ モード切り替えキー
- ⑭ 関数キー

(注) ● キャリッジリターンキーはどちらのキーを押しても同じです。
● スペースキー “ ” は本書では  と記載します。
● A C アダプター (E A -23 E) は別売品です。

基本操作とモードについて

本機には多くのキーがあります。各キーにはいろいろな文字や数字、記号が書かれています。
ここでは、これらの操作のしかた、モード、表示について簡単に説明します。

1. 電源のオン／オフ(入／切)と表示の濃度調整

(1) 電源のオン／オフ (入／切)

BREAK
ON (本体右上) を押してください。電源が入って次の表示 (画面) になります。

```

RUN MODE                                RUN
>                                         CAPS
                                         DEG
BUSY
WAIT
  
```

RUNモード (ランモード: 計算やプログラムを実行できる状態) になっていることを示します。
>マークはプロンプト記号といい、本機の準備が完了したことを示しています。

OFF (**ON** の左) を押せば電源が切れます。

電源を入れたり切ったりしたときに、一瞬、画面が黒くなったり不要な点や線またはシンボルが点灯することがありますが、機能などには問題ありません。

● オートパワーオフ機能について

電池の消耗を少なくするため、計算実行中 ("BUSY"シンボル点灯中) 以外の状態で、約11分間新たなキー操作を行わないと、自動的に電源が切れます。**ON** を押せば電源が入ります。

(2) 表示の濃度調整

表示が見やすくなるように表示濃度 (コントラスト) を調整してください。

SHIFT を押したまま コントラスト **ANS** を押します。

表示濃度の調整は、いずれかのモード (18ページ参照) の初期画面で行ってください。

```

*** LCD CONTRAST ***
      ▲    DARK
      ▼    LIGHT
  
```

▲を押すと濃くなり、▼を押すと薄く (淡く) になります。

● 濃度調整が終わったら、**BASIC**、**TEXT** などのモードキーを押します。

(次の「2. モードについて」を参照してください。)

2. モードについて

本機には、大きく分けて次の7つのモード (状態) があります。

RUNモード (実行モード) : **BASIC** プログラムを実行したり、マニュアル操作 (手操作) での計算などを行います。

PROモード (プログラムモード) : **BASIC** プログラムの入力、編集などを行います。

TEXTモード (テキストエディタモード) : テキストプログラムの入力、編集、削除、また **SIO** などへの出力、読み出し、**BASIC** プログラムとテキストプログラムの変換、ソースプログラムの入力などを行います。

ASMBLモード (アセンブラモード) : 機械語のアセンブルを行います。

CASLモード (キャッスルモード) : **CASL** のソースプログラムのアセンブル、実行を行います。

PICモード (ピックモード) : **PIC** 用プログラムの作成などを行います。

C言語モード : C言語のコンパイルや実行を行います。

モードの切り替えは右上にある緑色の **BASIC** と **TEXT** のキーと左端にある **SHIFT** および **A**、**C**、**P** で行います。

指定されているモードを表すシンボル (**RUN**、**PRO**、**TEXT**、**CASL**) が画面の右側に点灯します。**ASMBL**モード、**C言語モード**および**PICモード**を表すシンボルは存在しません。

モードの切り替え

設定するモード	キ ー 操 作
RUNモード	BASIC
PROモード	BASIC または BASIC BASIC
TEXTモード	TEXT
ASMBLモード	SHIFT + ASMBL を押してから A
CASLモード	SHIFT + ASMBL を押してから C
PICモード	SHIFT + ASMBL を押してから P
C言語モード	SHIFT + TEXT

← **RUN**モード以外から切り替えるときは **BASIC** を2回押す。

● たとえば **SHIFT** + **ASMBL** は **SHIFT** を押したまま ASMBL **BASIC** を押すことを示します。なお、この場合 **2nd F** **ASMBL** と押しても同じです。

● 統計モードについては62ページを参照ください。

● 機械語モニタモードについては274ページを参照ください。

3. 基本的なキー操作

それでは、アルファベットや記号を入力してみましょう。

いったん **[OFF]** を押して電源を切ってから、改めて **[ON]** を押して電源を入れてください。RUNモードになります。そして、**[CLS]** を押して表示内容を消去し、次のようにキーを押してみてください。キーを押しまちがえたときは、**[CLS]** を押して、最初から操作してください。

(1) アルファベット（大文字）の入力

次のキーを押してください。

[A] [B] [C] [D] [E] [F] [G]

ABCDEF G _

このように、アルファベットの大文字が表示されます。

ABCDEF Gと表示されないときは、次の確認をしてください。

- ①画面右側に“CAPS”シンボルが点灯していますか。点灯していないときは**[CAPS]**を押して点灯させてください。
- ②画面右側に“カナ”シンボルが点灯していませんか。点灯していたら**[カナ]**を押して消してください。

(2) アルファベット（小文字）の入力

次にアルファベットの小文字を入力してみましょう。

[CAPS] を押して画面右側の“CAPS”シンボルを消して、次のキーを押してください。

[H] [I] [J] [K] [L] [M] [N]

ABCDEF G h i j k l m n _

このように、“CAPS”シンボルを消して、アルファベットキーを押せばアルファベットの小文字が表示されます。再び大文字を入力する場合は、“CAPS”シンボルを点灯させてください。

(3) 記号の入力

次に、# \$ % など、キーの上側に橙色で書かれている記号を入力してみましょう。

これら、橙色で書かれている記号を入力するときは、次の2つの方法があります。

- ① **[SHIFT]** を押したまま、これらの記号が書かれているキーを押す。
- ② これらの記号が書かれているキーを押す前に **[2nd F]** を押す。

まず、**[SHIFT]** を押したまま、次のキーを押してください。

[E] [R] [T]

ABCDEF G h i j k l m n # \$ % _

次に、各キーを押す前に **[2nd F]** を押してください。

[2nd F] [I] [2nd F] [O] [2nd F] [P]

ABCDEF G h i j k l m n # \$ % < > @

(**[2nd F]** を押したとき、画面右側に“2nd F”シンボルが点灯します。)

このように橙色で書かれている記号を入れるとき、または橙色で書かれている機能を使うときは、**[SHIFT]** を押したまま、それぞれのキーを押すか、それぞれのキーを押す前に、**[2nd F]** を押します。

次に数字を入れてみましょう。

[1] [2] [3] [4] [5] [6]
[7] [8] [9]

ABCDEF G h i j k l m n # \$ % < > @ 1 2 3 4
5 6 7 8 9 _

- 1行24桁を超えると、次の行に入ります。



入力について

キーを押して、数値や文字を計算機に記憶させることを「入力」や「置数」などといいます。

(4) カーソルについて

次に、**[◀]** を1回押してください。すると“9”のところで **[▶]** マークが点滅します。

ABCDEF G h i j k l m n # \$ % < > @ 1 2 3 4
5 6 7 8 9

次に **[▶]** を押したままにしてみます。

しばらくすると、**[▶]** マークが左に移動していきます。左端まで行くと上の行の右端へ移り、再び左へ移動します。1行目の左端で止まります。

次に **[▶]** を押したままにします。**[▶]** マークが右に移動していき、最後に **[▶]** マークが消えて **[▶]** マークが点灯します。

この **[▶]** マークと **[▶]** マークは、カーソルといわれるもので、キーを押したときに文字などが入る位置を示しています。

キーを押しまちがえて、違う文字や記号が入ってしまったときは、カーソルをまちがった文字などの位置に移動させて、正しいキーを押せば訂正できます。(くわしくは26ページを参照してください。)

こんどは **[▲]** を押してみてください。カーソルが1行上(1行目)へ移動します。

次に **[▼]** を押してみてください。カーソルが1行下(2行目)へ移動します。

このように、**[▲]** **[▼]** はカーソルを上下に動かします。

カーソルが先頭行にあるときに **[▲]** を押すと、その行の先頭（左端）へ移動します。
 カーソルが最後の行にあるときに **[▼]** を押すと、その行の最後へ移動します。

キーの記載方法について

- この取扱説明書では、キー操作またはキーの機能を次のように記載します。

ASMBL
[BASIC] の例

- ①キーに書かれている機能を使用する場合は、単に **[BASIC]** と記載します。
- ②キーの上側に橙色で書かれている機能を使用する場合は、**[SHIFT]** + **[ASMBL]** または **[2nd F]** **[ASMBL]** と記載します。

なお、**[SHIFT]** + **[ASMBL]** は **[SHIFT]** を押したまま **[ASMBL]** を押すことを意味します。

また、**[2nd F]** **[ASMBL]** は **[2nd F]** を押して離した後、**[ASMBL]** を押すことを意味します。

- ただし、数字やアルファベット、記号、カタカナなど、単にキーや本体（キーの上側）に書かれている文字や数字、記号を入力するだけの場合は、キーの枠や **[SHIFT]**、**[2nd F]** の操作などを省いて記載します。

〈例〉 **[S]** **[H]** **[A]** **[R]** **[P]** → SHARP

[A] **[S]** **[SHIFT]** + **[S]** → ASS

- スペースキーは **[SPACE]** と記載します。
- この計算機は、数字の 0 を \emptyset と表示します。

これは、アルファベットのオー（O）と区別するためです。本書でもオーとゼロの区別がつきにくい場合はゼロを \emptyset と表しています。

なお、説明上必要な場合には、CA
[CLS] などの方法で示しています。

4. 表示シンボルについて

本機には色々な機能やモードがあります。そして、いま本機がどのような状態になっているのかが分かるように、画面の左や右にシンボルを点灯させます。

次に表示シンボルとその意味を示します。

なお、本書の表示例では、説明上必要なシンボルだけを記載しています。



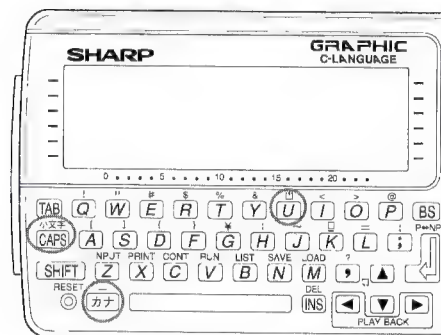
< シンボル >

< 説明 >

BUSY	: 計算実行中または命令実行中です。
BATT	: 電池が消耗していることを示します。このシンボルが点灯したときは、速やかに電池を交換してください。
RUN	: BASICのRUNモード（実行モード）になっていることを示します。 [BASIC] で点灯できます。
PRO	: BASICのPROモード（プログラムモード）になっていることを示します。 [BASIC] で点灯できます。
TEXT	: TEXTモード（テキストモード）になっていることを示します。 [TEXT] で点灯できます。
CASL	: CASLモード（キャスルモード）になっていることを示します。 [2nd F] [ASMBL] （または [SHIFT] + [ASMBL] ）と押してから [C] を押せば点灯できます。
STAT	: 統計モードになっていることを示します。 [2nd F] [STAT] （または [SHIFT] + [STAT] ）と押せば、点灯／消灯ができます。
2nd F	: [2nd F] が押されたことを示します。 このシンボルが点灯しているとき、各キーの橙色で書かれている機能が指定されていることを示します。
M	: マニュアル（手操作）による計算用のメモリに 0 以外の数値が入っていることを示します。
CAPS	: アルファベットの大文字が入力できる状態になっています。このシンボルが消えているときは、アルファベットの小文字が入力できる状態になっています。この状態は [CAPS] で切り替えます。 ただし、“カナ” シンボルが点灯しているときは、カナの入力が優先され、アルファベットは入力されません。
カナ	: ローマ字のつづりでキーを押せば、カタカナを入力できます。（23ページと382ページ参照） [カナ] を押すことにより、このシンボルの点灯／消灯ができます。
小	: 小さいカナ文字（フィウエオヤユョッ）を入力することができます。 “カナ” シンボル点灯時、 [CAPS] でこのシンボルを点灯できます。
DEG RAD	: 三角関数、逆三角関数、座標変換を行うときに使用する角度の単位を示します。（関数計算の項を参照）
GRAD	
CONST	: 定数が記憶されていることを示します。（定数計算が設定されていることを示します。36ページ参照） このシンボルが点灯しているときは、 [定数] で定数計算が行われます。 不要なときは [2nd F] [CA] （ [SHIFT] + [CA] ）と押して、定数を消去してください。
PRINT	: プリント（印字）モードになっていることを示します。 プリンタが接続されているときに、 [2nd F] [P↔NP] と押せば、点灯／消灯ができます。

5. カタカナの入力のしかた

本機にはローマ字のつづりで入力した文字をカナに変換する“ローマ字→カナ変換機能”があります。カナは左下にある **[カナ]** を押して、画面右側に“カナ”シンボルを点灯させ、アルファベットキーを用いて入力します。



[カナ]このキーで“カナ”シンボルの点灯／消灯を行います。“カナ”が点灯しているときに、カナを入力できます。

[CAPS]アイウエオヤユョツ（小さいカナ文字）を入力するときに使います。

[SHIFT] + [U]「ン」を入力するときに使用します。

<例>	(入力文字)	(キー操作)
カタカナ	→	KATAKANA
ガッコウ	→	GAKKOU
ヘンカン	→	HENKAN [SHIFT] + [U]
ハンイ	→	HAN [SHIFT] + [U] I
ディスク	→	DH I S U K U
ウォッチ	→	U [CAPS] O T T I

- 「ン」を入力するときに、**[N]**の後にYを除く子音がくる場合は **[SHIFT] + [U]** を押す必要はありません。
- ローマ字の表記の方法については、382ページを参照してください。

- カナ入力中は、押したキー（子音）が画面右に表示されます。

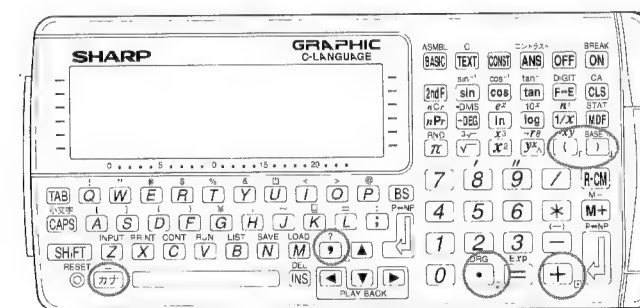
<例> S I N J Y



ジュを入力するために J Y と押したところでは、U を押すとカーソル位置にジュが表示され、この表示は消えます。この表示は1行目に出る場合もあります。

カナ記号の入力のしかた

カナ記号は次のキー操作により入力します。



- ー（長音符） : **[SHIFT] + [カナ]**
- 、（読点） : **[.]**
- 。（句点） : **[.]**
- ・（中点） : **[+]**
- 「（カギカッコ） : **[[]]**
- 」（カギカッコ） : **[)]]**

第1章

マニュアル計算（手操作による計算）

1. マニュアル計算

マニュアル計算とは、1つ1つの計算を1回1回手操作によって計算することをいいます。関数電卓として使用する場合とほぼ同じですが、BASICプログラムを組む前に、必ず知っておかなければならないことについて説明します。60進数の度・分・秒計算機能は、マニュアル計算でのみ可能です。

①計算式の作りかた

②まちがえたときの訂正のしかた

③連続計算のしかた

④関数の使いかた

など、基本的で重要な事項です。よく読んで理解し、本機の操作に慣れ、いろいろな基本計算をマスターしてください。

2. マニュアル計算のしかた

(1)電源オン

[ON] を押して、電源を入れます。



“RUN” シンボル[※]が点灯します。これはマニュアル計算ができることを示しています。

もし、RUNのシンボルが点灯していないときは**[BASIC]**を押してRUNを点灯させてください。

※説明上必要な場合を除き、「シンボル」の文字は省略します。

■計算の記号について

通常の数学では四則計算の記号として「+、-、×、÷」を使いますが、BASIC言語では「×」や「÷」の代りに「*」や「/」などの記号を用います。また、本機ではべき乗や指数、その他の関数などにおいて、通常の数学で用いられている記号とは違いかたちで表現されていますので、以下に本機のマニュアル計算やBASIC言語で使用する計算の記号や特殊記号について説明します。

	計 算	数学で用いる記号	本機で用いる記号	読 み か た
(1)	加 算	+	+	プラス
(2)	減 算	-	-	マイナス
(3)	乗 算	×	*	アスタリスク
(4)	除 算	÷	/	スラッシュ
(5)	整数の除算		¥	エン
(6)	整数の剰余		MOD	モッド
(7)	符 号	-または+	-, +	マイナス、プラス
(8)	計算の実行	=		キャリッジリターン

〔補足〕 割る数と割られる数の両方の、小数点以下第1位を四捨五入して、整数化してから除算を行います。¥では商が求まり、MODでは余りが求められます。

〔例〕 $51 \div 5 \rightarrow 10$ $51 \text{ MOD } 5 \rightarrow 1$ $(51 \div 5 = 10 \dots 1)$
 $51 \div -5.7 \rightarrow -8$ $51 \text{ MOD } -5.7 \rightarrow 3$ $(51 \div -5.7 = -8 \dots 3)$
 $87.57 \div 5.4 \rightarrow 17$ $87.57 \text{ MOD } 5.4 \rightarrow 3$ $(87.57 \div 5.4 = 17 \dots 3)$
 $30^\circ 36' \div 14^\circ 36' \rightarrow 2$ $30^\circ 36' \text{ MOD } 14^\circ 36' \rightarrow 1$ $(31 \div 15 = 2 \dots 1)$
 $30^\circ 36' \div 4.4 \rightarrow 7$ $30^\circ 36' \text{ MOD } 4.4 \rightarrow 3$ $(31 \div 4 = 7 \dots 3)$
 $87.57 \div 14^\circ 36' \rightarrow 5$ $87.57 \text{ MOD } 14^\circ 36' \rightarrow 13$ $(88 \div 15 = 5 \dots 13)$

3. キー操作の練習（訂正のしかた）

ここで簡単な例題をもとに、キー操作の練習をします。

数値などを入れているときに違った数字や命令を入れたり、余分な数字や命令を入れたり、あるいは抜かしてしまったときの訂正のしかたです。

〔例題〕①

計 算 式	キ ー 操 作	答
$5 + 15 \times 2 \div 4$	$5 \text{ [+] } 15 \text{ [*] } 2 \text{ [/] } 4 \text{ [=]}$	12.5

(1)まちがって他のキーを押したとき

$5 + 15 \times 2 \div 4$ を $5 + 15 \times 2 \times 4$ とした場合 	
キ ー 操 作	表 示 部
$5 \text{ [+] } 15 \text{ [*] } 2 \text{ [*] } 4$	$5 + 15 * 2 * 4$ ↑ ここがちがう
[◀ ▶] このキーを押して左へ（2回押す）カーソル*を移動させる	$5 + 15 * 2 * 4$ ↑ この部分にカーソルを移動させて点滅させる
[/] (この後 [▶▶] としてもよい)	$5 + 15 * 2 / 4$ ↑ ↑ 正しいキーを入れる ここが点滅する

	このキーを押して右へカーソルを移動させる	$5 + 1.5 \times 2 \div 4$
		12.5 答
キー入力した式を確認したいとき（これをプレイバック機能という）		
	このキーを押すと	$5 + 1.5 \times 2 \div 4$
	を押して	12.5
	さらにこのキーを押すと	$5 + 1.5 \times 2 \div 4$ ↑ この部分が点滅する
		12.5

※カーソルについて

キー操作を行うとき表示部に が表示されたり、キー操作の訂正のときに マークが点滅したりしますが、これらのマークはカーソルと呼ばれるもので、次に操作したキーがこの部分に入ること示しています。つぎに説明する挿入モードでの もカーソルです。

■まちがって余分なキーを押したとき

〔例題〕②

計 算 式	キ ー 操 作	答
$20 \times 3 \div 5$	$20 \times 3 \div 5$	12

キ ー 操 作	表 示 部
$20 \times 3 \times \div 5$	$20 \times 3 \times \div 5$ ↑ 余分なキー ↑ 数字の0（ゼロ）と英文字のO（オー）とを区別するために 0（ゼロ）→ \emptyset と表示する
このキーを3回押す	$20 \times 3 \times \div 5$ ↑ この部分にカーソルを移動させる
(SHIFT) + (DEL) (下記の※を参照)	$20 \times 3 \div 5$ ↑ * が削除されてここが点滅する
	12.
キー入力した式を確認したいとき 	$20 \times 3 \div 5$

※ **(SHIFT) + (DEL)** は **(SHIFT)** を押したまま **(INS)** を押すことを示します。なお、**(2nd F) (INS)** と押しても DEL 機能（削除）が働きます。

(3)プレイバック機能について

〔例題〕②において、余分なキーを押したのに気がつかずに を押してしまうと、エラーが発生します。このようなとき、キー入力した式を呼び戻して訂正する機能がプレイバック機能です。この機能を使うとエラーを起こした場所をカーソルで示しますので、エラーの原因がわかりやすく大変便利です。

キ ー 操 作	表 示 部
$20 \times 3 \div 5$	ERROR 10
	$20 \times 3 \div 5$ ↑ ここが点滅する 例題の式は $20 \times 3 \div 5$ なので この場合 * が不要です
(BS) (この後 としてもよい)	$20 \times 3 \div 5$ ↑ カーソルの前の * が削除されます
2回押して	$20 \times 3 \div 5$ ↑ カーソルが最後に表示されます
	12.

(4)キー操作が抜けたとき

〔例題〕③

計 算 式	キ ー 操 作	答
$2 \times 3 \div 10$	$2 \times 3 \div 10$	0.6

キ ー 操 作	表 示 部
$23 \div 10$	$23 \div 10$ ↑ ここに * が抜けている
 カーソルを左へ4つ移動	$23 \div 10$ ↑ この部分を点滅させます
(INS) 訂正モードから挿入モードに変わります	$23 \div 10$ ↑ カーソルが <u> </u> から <u> </u> に変わり、次に入力した文字がカーソルの前に挿入されます
(*) (この後 としてもよい)	$2 * 3 \div 10$ ↑ * が挿入（追加）されます
 カーソルを右へ4つ移動	$2 * 3 \div 10$
	0.6

(5)数式において誤りが多いとき

(CLS) を押して、今まで入れた内容をすべて消し、あらためて最初から入力し直します。

(6)DELとINS、BSの意味

DEL → DELETE……（デリート）削除を意味します。

INS → INSERT……（インサート）挿入を意味します。

入力した文字がカーソルの前に挿入されます。再度 **(INS)** を押すと訂正

モードに変わります。また、を押したときも挿入モードは解除され訂正モードに戻ります。

BS → BACKSPACE……（バックスペース）1文字分の後退を意味します。
カーソルの前の文字が削除されます。

練習問題 1

を行ってください。

練習問題は74ページから記載しています。















4. エラーの処理について

練習問題 1 を行って、すぐに答えの出た人もいるでしょう。また、答えが出ずにERROR 1 0 などという表示が出た人もいることでしょう。

ERROR（エラー）が出た人も出なかった人も、次の例題でエラーの処理を行ってみてください。

〔例題〕④

$$\frac{36}{2+4}$$

キ ー 操 作	表 示 部
3 6  2  4 	3 6 / 2 + 4) _
	ERROR 1 0
	3 6 / 2 + 4) ↑ ここが点滅します この意味は  (カッコ)の部分がエラーの原因であることを示しています
この〔例題〕④の場合は36 / (2 + 4) が正しい式ですから	
  	3 6 / 2 + 4) ↑ ここが点滅します
 INS 挿入モードにします	3 6 / 2 + 4)
	3 6 / (2 + 4)
	6 .
式を確認したいときは 	3 6 / (2 + 4) _ ↑ カーソルが最後に表示されます
または 	3 6 / (2 + 4) ↑ カーソルが先頭に表示されます



() について

分数においては、式にカッコがついていなくても、分母、分子はそれぞれ全体をカッコでくくる必要があります。ただし、分母、分子とも数値が1つだけの場合は省略できます。

なお、関数キーを使う場合（45ページ参照）も中カッコ { } 等を必要とする場合があります。






計算機では小カッコも中カッコも同じ   のキーを使います。



5. 数式が長い場合の連続計算のしかた

〔例題〕⑤

$$2 \times 3 \div 10 \times 100$$

↑
ここでひとまず区切る


キ ー 操 作	表 示 部
2  3  1 0 	0 . 6
↑ これで区切り計算となります	
 1 0 0	0 . 6 * 1 0 0 _
	6 0 .


計算式の途中でを押すと、それまでの数値計算が実行されるので、区切りのよいところでを押します。続いて計算記号と数値をキー入力すれば、長い数値の計算式でも計算が容易になります。式が表示上で255字分以内であれば、区切ることなく計算できます。

練習問題 2

を行ってください。

6. ラストアンサー機能について

これは、計算した結果を記憶している機能です。具体的にはマニュアル計算でを押して得られた結果や、ダイレクトアンサー機能を用いて得られた結果（これらをラストアンサーと呼ぶ）などを記憶します。この機能は、次のような使いかたをすると便利です。

ダイレクトアンサー機能：  を使用せずに関数計算を行う方法。

数値を入れてから、各関数キーを押す。（38ページ参照）

〔例題〕⑥

$$\frac{4 \times 6}{4 + 6} = 2.4 \quad \text{この } 2.4 \text{ という結果を使って}$$

$$\frac{1}{2.4} + \frac{3}{2.4} + \frac{2}{2.4} \quad \text{のような計算を行う場合}$$

キ ー 操 作	表 示 部
4 [×] 6 [÷] 4 [+] 6 [÷] [↵]	2.4
1 [÷] [ANS] [+] 3 [÷] [ANS] [+]	1 / 2.4 + 3 / 2.4 + _
2 [÷] [ANS]	1 / 2.4 + 3 / 2.4 + 2 / 2.4 _
[↵]	2.5
ここで [↵] を入れたことによって、新しい計算を実行したことになりますのでラストアンサーは、2.5という数値に入れ替わります。	

ラストアンサー機能とは、ひとつの記憶箱（レジスタ）にひとつのデータをしまっておくことであり、常に一番新しいデータだけが記憶できます。

なお、ラストアンサーはPROおよびRUNモードのとき、[ANS] で呼び出すことができます。

7. 計算結果の表示方法

いろいろな計算を実行する中で、小数部については小数点以下2桁、あるいは3桁くらいで十分という場合が多いものです。また、小数点の表示ではなく、指数での表示のほうが便利でわかりやすい場合もあります。

そこで、計算結果の表示方法について、本機の便利な利用法を説明します。

通常用いている数値は、

12300や0.0987のように表されます。

しかし、理科や工科系においては、

12300を 1.23×10^4

0.0987を 9.87×10^{-2}

と、表すことが多くあります。

数値をこのように表す方法を**指数方式**と呼びます。

1.23×10^4

9.87×10^{-2}

この部分を**指数部**といいます。

この部分を**仮数部**といいます。

本機では、数値を指数方式で表示する場合は、指数部を示す記号としてE（[2ndF] [Exp] で入力）を用います。次の例題でキー操作の練習をしてください。

キ ー 操 作	表 示 部
例題A $1.2 \times 10^9 =$ 1 [.] 2 [2ndF] [Exp] 9 [↵] [F↔E] 普通の表示を指数表示にします	1.2E9_ 1200000000. 1.2E09
例題B $1.2 \times 10^{10} =$ 1 [.] 2 [2ndF] [Exp] 10 [↵] [F↔E]	1.2E10 1.2E10 ここが2桁になると[F↔E]を押しても変わりません
例題C $0.0987 =$ 0 [.] 0987 [↵] [F↔E]	0.0987 9.87E-02 10 ⁻² を意味します
例題D $\frac{1.71 \times 2.43 \times 10^3}{3.25 \times 1.5 \times 10^{-2}} =$ 1 [.] 71 [×] 2 [.] 43 [2ndF] [Exp] 3 [÷] 3 [.] 25 [×] 1 [.] 5 [2ndF] [Exp] -2 [÷] [↵] [F↔E]	85236.92308 8.523692308E04

〔注〕 $a \times 10^x$ を a [2ndF] [Exp] x と置く場合、x は ±(0 ~ 99) の整数でなければなりません。整数3桁以上の数値を入れたときは、下から2桁をxと読んでしまいます。また、小数部分をもった数値を入れたときは、ERROR10になります。

〈例〉 1 [2ndF] [Exp] -1125 [↵] → 1.E-25 (誤り)

53ページ「[2ndF] [10^x]、[2ndF] [e^x]、[2ndF] [Exp] の違いについて」を参照。



大きな数字、小さな数字について

この計算機は10桁の計算機です。値の大きな数字（指数部が10以上）や小さな数字（指数部が-10以下）は通常の表示方法では表示できません。そのため、[F↔E] を押しても表示は変わらず、常に指数方式で表示されます。

8. 表示に関するフォーマット指定

(1) DIGIT指定

マニュアル計算で得られた結果などでは、特殊計算を除いて小数部は2桁や3桁などよい場合が多いものです。そこで、計算する前あるいは計算結果が表示されているときにDIGIT（デジット）によって計算結果の小数部の桁数を指定することができます。

- 〈例〉 $\text{2nd F} \text{ DIGIT}$ 0 整数を表示 (小数点以下1桁目を四捨五入)
 $\text{2nd F} \text{ DIGIT}$ 2 小数点以下2桁まで表示 (小数点以下3桁目を四捨五入)
 $\text{2nd F} \text{ DIGIT}$ 4 小数点以下4桁まで表示 (小数点以下5桁目を四捨五入)
 $\text{2nd F} \text{ DIGIT}$ \square デジット指定解除 電源オフやモードの切り替えなどでも解除

(注) ● 結果が60進数の度・分・秒の記号で表示されたときは、DIGIT指定は無効です。

〔例題〕⑦

$\frac{1.71}{3.5}$ の計算を小数点以下2桁まで求めなさい。

キ ー 操 作	表 示 部
まず、デジット指定で小数部の桁数を指定します。 $\text{2nd F} \text{ DIGIT}$ 2	
$1 \square 71 \square 3 \square 5 \square$	0.49
1.71/3.5の結果は0.488571428ですが小数部分を2桁に指定していますので、3桁目が四捨五入されて0.49となります。	

■ USING 命令

USING (ユージング) 命令でフォーマットの指定をするときは、計算の前にあらかじめ指定しておく必要があります。最後の桁は切り捨てられた値となります。

- ① USING '####' 符号と整数2桁を表示
 ② USING '####.' 符号と整数2桁と小数点を表示
 ③ USING '####.###' 符号と整数2桁と小数点と小数点以下2桁を表示
 ④ USING '####.####.' 符号、3桁区切りマーク(,)、整数5桁と小数点を表示
 3桁区切りマークも1桁と数えます。たとえば-1,234,567.を表示させるときは、USING '#####.###.'のように#と、をあわせて最低10個指定する必要があります。
 ⑤ USING '###.##^' 小数点以下2桁までの指数方式で表示
 このとき指数部はEと符号を含めて4桁(E-00)が、自動的に取られます。
 ⑥ USING フォーマット指定を解除

(注) ● 符号が正のときはスペースになります。

- USING 命令を実行すると、デジット指定は解除されます。
同様にデジット指定により、USINGの指定は解除されます。
- 結果が60進数の度・分・秒の記号で表示されたときは、USING命令は無視されます。

〔例題〕⑦をUSINGを使用して計算してみましょう。

キ ー 操 作	表 示 部
$\text{U} \text{ S} \text{ I} \text{ N} \text{ G}$ (SHIFT) + \square (SHIFT) + \square	USING '##_
(SHIFT) + \square (SHIFT) + \square	USING '###.##_
(SHIFT) + \square (SHIFT) + \square	USING '###.###^_
\square	>
$1 \square 71 \square 3 \square 5 \square$	0.48
	これは「+0.48」ということを意味しています
またここで (ANS) を押すと	4.885714286E-01_
そしてさらに \square を押すと	USING指定でない表示がでます
	0.48
	USING指定に戻ることができます

- 計算が終わったら、 $\text{U} \text{ S} \text{ I} \text{ N} \text{ G}$ \square と押してUSING指定を解除してください。

練習問題 3

を行ってください。

9. 数値丸め機能 MDF (モディファイ)

次の計算をしてみましょう。

$$\frac{55.4}{9} \times 9$$

キ ー 操 作	表 示 部
(2nd F CA)	
$55.4 \square 9 \square$	6.155555556
$\square * 9$	6.155555556 * 9_
\square	55.4

この計算例では小数部がたくさん表示されるのですっきりせず、わずらわしく感じます。そこで有効桁数を考え、式の途中の結果の端数を四捨五入し、しかも、最終結果においてより確からしい値を求める方法があります。これを数値丸め機能 (MDF) といいます。

先の例において

$$55.4 \div 9 = 6.155555556 \text{ ですが、小数第4位を四捨五入し } 6.156 \text{ として}$$

$$6.156 \times 9 \text{ を計算すれば } 55.404 \text{ となります。}$$

キ ー 操 作	表 示 部
$\text{2nd F} \text{ DIGIT}$ 3	
$55.4 \square 9 \square$	6.156
(MDF)	6.156
$\square * 9$	6.156 * 9_
\square	55.404

このように数値丸め機能を用いれば、計算途中で有効桁数の端数処理を行いながらより確からしい値を求めることができます。

ただし、この機能(MDF)はDIGITまたはUSINGの指定がされているときだけ有効です。

(注) ● 結果が60進数の度・分・秒の記号で表示されたときは、この機能は無効です。

10. 計算結果の符号の反転

計算によって得られた結果を次の計算に利用する場合、その結果の符号を反転させてから使用したいときに便利な機能です。

たとえば、1572円、2350円、3058円の買い物をして10000円で支払う場合、買い物の合計金額とお釣りの金額を求めるときの操作は次のようになります。

キ ー 操 作	表 示 部
1572 (+) 2350 (+) 3058 (=)	6 9 8 0 .
(2nd F) (-)	- 6 9 8 0 .
(+) 10000 (=)	3 0 2 0 .
(注) $A - B = -B + A$ により $10000 - 6980$ を $-6980 + 10000$ として計算	
または	
1572 (+) 2350 (+) 3058 (=)	6 9 8 0 .
(-) 10000 (=)	- 3 0 2 0 .
(2nd F) (-)	3 0 2 0 .

11. メモリ計算

マニュアル計算で、数値を一時記憶したり、計算結果を累計できるメモリがあります。

このメモリは次のキー操作で利用できます。

(R・CM) メモリの内容を表示します。

続いてもう1回押せばメモリの内容を消去します。メモリ計算は、このキー操作でメモリ内を消去してから始めます。

(M+) 表示されている数値をメモリに加えます。また、式が表示されているときは、その式を計算して、結果をメモリに加えます。

(2nd F) (M-) 表示されている数値をメモリから減算します。また、式が表示されているときは、その式を計算して、結果をメモリから減算します。



メモリについて

メモリとは数値(お金)をたくわえておく金庫のようなものです。金庫に次々と数値を追加したり(M+)、必要な分だけ引き出したり(2nd F) (M-) することができます。また、金庫にたくわえられている値(全額)を呼び出す(R・CM) こともできます。

R・CMとはRM (Recall Memory メモリの呼び出し) とCM (Clear Memory メモリの消去) のことです。この2つの働きを1つのキーにまとめて表しています。

〔例題〕⑧

(1) 24×34 、 $64 \div 5$ 、 $13 \times 28 \div 8$ の計算を行い、その結果の合計を求めなさい。

(2) 次の計算を行いなさい。

$$24 + 46 =$$

$$- \rightarrow 37 \times 4 =$$

$$+ \rightarrow 84 - 29 =$$

合計

キ ー 操 作	表 示 部	備 考
(R・CM) (R・CM) (CLS)		メモリ内容を消去
24 (*) 34 (M+)	8 1 6 .	計算結果をメモリに加えます。
64 (/) 5 (M+)	1 2 . 8	このとき、メモリが使われていることを示す M シンボルが点灯します。
13 (*) 28 (/) 8 (M+)	4 5 . 5	合計が求められます。
(R・CM)	8 7 4 . 3 _	
(R・CM) (R・CM) (CLS)		メモリ内容を消去 (M シンボル消灯)
24 (+) 46 (M+)	7 0 .	{ 37×4 の計算を実行し、その結果をメモリから減算します。(※参照)
37 (*) 4 (2nd F) (M-)	1 4 8 .	
84 (-) 29 (M+)	5 5 .	
(R・CM)	- 2 3 . _	合計を呼び出します。

※次のように操作することもできます。

37 (*) 4 (=) 37×4 を計算し、結果を表示

(2nd F) (M-) 表示されている数値をメモリから減算

12. 定数計算機能

定数計算とは、ある決まった値(定数)に対して決まった計算を行うことです。次のキーにより定数と計算を設定できます。

(CONST) 定数と計算を設定

(1) 設定方法 (注) a は定数とする数値(または式)を示します。

加算: (+) a (CONST) (+ a が定数) または a (+) (CONST) (a + が定数)

減算: (-) a (CONST) (- a が定数) または a (-) (CONST) (a - が定数)

乗算: (*) a (CONST) (× a が定数) または a (*) (CONST) (a × が定数)

除算: (/) a (CONST) (÷ a が定数) または a (/) (CONST) (a ÷ が定数)

定数計算を設定すると表示部右側に "CONST" が点灯します。

(2) 設定内容の確認

"CONST" が点灯しているときに次のキーを押せば四則の記号(+ - * /) も含めて設定内容が表示されます。

(2nd F) (CONST) (SHIFT) + (CONST)

(3) 設定内容の消去

次のキー操作で設定内容が消去されます。また、電源を切ったときも消去されます。

$\text{[2nd F] [CA]} (\text{[SHIFT]} + \text{[CA]}) \dots\dots\dots$ “CONST” が消灯します。

〔例題〕⑨

- (1) $+(4.8+3.6)$ を定数とする、次の計算を行いなさい。
 $32.5 + (4.8 + 3.6) =$
 $24 - 18.5 + (4.8 + 3.6) =$
 $8.2 \times 6 + (4.8 + 3.6) =$
- (2) $(57-14) \times$ を定数とする、次の計算を行いなさい。
 $(57-14) \times 18 =$
 $(57-14) \times (27 \div 4) =$
 $(57-14) \times (24+43) =$

キ ー 操 作	表 示 部	備 考
$\text{[+] [C] 4.8 [+] 3.6 [C]}$ [CONST]	$+ (4.8 + 3.6) _$ >	定数計算設定 (“CONST” 点灯)
$32.5 [\text{↵}]$ $24 [-] 18.5 [\text{↵}]$ $8.2 [*] 6 [\text{↵}]$ [2nd F] [CONST]	4 0 . 9 1 3 . 9 5 7 . 6 $+ 8 . 4 _$	設定内容確認 (4.8+3.6の結果が定数になっています)
$\text{[C] 57 [-] 14 [C] [*]}$ [CONST]	$(57 - 14) * _$ >	定数計算設定 (前の設定内容は消されます)
$18 [\text{↵}]$ $27 [\div] 4 [\text{↵}]$ $24 [+] 43 [\text{↵}]$ [2nd F] [CA]	7 7 4 . 2 9 0 . 2 5 2 8 8 1 . >	設定内容消去

第2章

関数計算の操作方法と練習

1. マニュアル (手操作) での関数計算

マニュアルにおいて、関数計算を単独で行う場合の方法をここで説明します。

本機には、BASIC言語でも使える基本関数がありますが、それらについてはBASICプログラムの項で説明します。

マニュアル計算はRUNモードで行います。 [BASIC] で画面の右上に“RUN”を点灯させてください。

PROモードでマニュアル計算はできません。

60進数の度・分・秒計算機能は、マニュアル計算でのみ可能です。

関数キーを使う方法

マニュアル計算の場合は、通常、関数キーを使います。

RUNモード

- ① 関数キーを押して、続いて数値を入れ [↵] を押す。
- ② 数値を入れ、続いて関数キーを押す。
 ②の場合、関数電卓と同じような使いかたとなり、 [↵] を押さなくても計算結果が出ます。これをダイレクトアンサー機能といいます。ただし、この機能は、べき乗や座標変換のように2つの数値が必要な関数、また式の入力途中では働きません。
 表の中にこのダイレクトアンサー機能を(DAns 機能)として示しています。

アルファベットキーを使う方法

関数キーを用いなくても、アルファベットで表現されるものは、アルファベットキーで入力することもできます。関数キーが用意されていない関数はアルファベットキーで入力します。

たとえば、

5^2 の計算では、 $\text{[x}^2\text{] 5}$ と操作しても、 [S] [Q] [U] 5 と操作しても答は同じです。プログラムの中で関数を扱う場合は、アルファベットの使用頻度が高いため、この方法が便利なときがあります。

(1)関数入力の方法 ①

(注) DAns: ダイレクトアンサー

項 目	通常の表現	本機の表現	読 み か た	入力方法と備考
2 乗	x^2	S Q U	スクウェア square (平方)	$[x^2]$ X $[=]$ または X $[x^2]$ (DAns 機能)
平方根	\sqrt{x}	S Q R	ルート root square root (平方根)	$[\sqrt{\quad}]$ X $[=]$ または X $[\sqrt{\quad}]$ (DAns 機能)
3 乗	x^3	C U B	キュービック cubic (立方)	$[2nd F]$ $[x^3]$ X $[=]$ または X $[2nd F]$ $[x^3]$ (DAns 機能)
立方根	$\sqrt[3]{x}$	C U R	キュービックルート cubic root (立方根)	$[2nd F]$ $[\sqrt[3]{\quad}]$ X $[=]$ または X $[2nd F]$ $[\sqrt[3]{\quad}]$ (DAns 機能)
逆 数	$\frac{1}{x}$	R C P	レシプロカル reciprocal (逆数)	$[1/x]$ X $[=]$ または X $[1/x]$ (DAns 機能)
円 周 率	π	P I	パイ Pi (ギリシャ語・円周率)	$[\pi]$ または $[P]$ $[I]$ $[=]$ $\pi \approx 3.141592654$
べき乗 (べき乗根)	y^x (注) $\sqrt[x]{y} = y^{\frac{1}{x}}$	^	ハットマーク power (累乗, べき乗)	Y $[y^{\wedge}]$ X $[=]$ (注) Y $[y^{\wedge}]$ $[\square]$ 1 $[\square]$ X $[=]$ $[=]$
絶 対 値	$ x $	A B S	アブソリュート absolute (絶対値)	$[A]$ $[B]$ $[S]$ X $[=]$ Xの絶対値
整 数 化		I N T	インテジャー integer (整 数)	$[\square]$ $[N]$ $[T]$ X $[=]$ X以下でかつ最も大きい整数
符号関数		S G N	シグナム sign (符 号)	$[S]$ $[G]$ $[N]$ X $[=]$ Xの値が正のとき1、負のとき-1、 0のとき0が得られる

(注)・「入力方法と備考」欄のX、Y、n、rは「通常の表現」欄のx、y、n、rに対応する数値を表します。また、 θ は角度を表す数値です。

・Xについては、60進数の度・分・秒の記号で指定することもできます。

(2)関数入力の方法 ②

項 目	通常の表現	本機の表現	読 み か た	入力方法と備考
乱 数		R N D	ランダム random numbers (乱 数)	$[2nd F]$ $[RND]$ X $[=]$ 乱数を発生する ● Xについては114ページを参照ください。
数 値 丸め機能		M D F	モディファイ modify (修正する)	$[MDF]$ X $[=]$ DIGIT またはUSING 命令で指定された桁数の端数処理を行う。
桁数指定		D I G I T	デジット digit	$[2nd F]$ $[DIGIT]$ n nは指定する小数点以下の桁数 $0 \leq n \leq 9$ 解除は $[2nd F]$ $[DIGIT]$ $[\square]$ と操作
階 乗	$n!$	F A C T	ファクトリアル factorial (階 乗)	$[2nd F]$ $[n!]$ n $[=]$ または n $[2nd F]$ $[n!]$ (DAns 機能)
順 列	$nPr = \frac{n!}{(n-r)!}$	N P R	パーミュテーション permutation (順 列)	$[nPr]$ $[\square]$ n $[\square]$ r $[\square]$ $[=]$ ただし、 $n \geq r$
組み合わせ	$nCr = \frac{n!}{r!(n-r)!}$	N C R	コンビネーション combination (組み合わせ)	$[2nd F]$ $[nC_r]$ $[\square]$ n $[\square]$ r $[\square]$ $[=]$ ただし、 $n \geq r$
対数関数	$\ln x$ または $\log_e x$	L N	ローン natural logarithm (自然対数)	$[\ln]$ X $[=]$ または X $[\ln]$ (DAns 機能)
	$\log x$	L O G	ログリズム common logarithm (常用対数)	$[\log]$ X $[=]$ または X $[\log]$ (DAns 機能)
指数関数	e^x (自然指数)	E X P	エクスポネント exponent (指数)	$[2nd F]$ $[e^x]$ X $[=]$ または X $[2nd F]$ $[e^x]$ (DAns 機能) $e \approx 2.718281828$
	10^x (常用指数)	T E N	テン power of ten	$[2nd F]$ $[10^x]$ X $[=]$ または X $[2nd F]$ $[10^x]$ (DAns 機能)

(3)関数入力の方法 [3]

項 目	通常の実現	本機の実現	読 み か た	入力方法と備考
三角関数	$\sin \theta$	S I N	サイン <u>sine</u> (正弦)	<ul style="list-style-type: none"> ● $\text{2nd F} \text{ DRG}$ で DEG、RAD、GRAD のいずれかを指定 $\text{sin } \theta \text{ } \blacktriangleright$ または $\theta \text{ sin}$ (DAns 機能)
	$\cos \theta$	C O S	コサイン <u>cosine</u> (余弦)	$\text{cos } \theta \text{ } \blacktriangleright$ または $\theta \text{ cos}$ (DAns 機能)
	$\tan \theta$	T A N	タンジェント <u>tangent</u> (正接)	$\text{tan } \theta \text{ } \blacktriangleright$ または $\theta \text{ tan}$ (DAns 機能)
逆三角関数	$\sin^{-1}x$	A S N	アークサイン <u>arc sine</u> (逆正弦)	<ul style="list-style-type: none"> ● $\text{2nd F} \text{ DRG}$ で DEG、RAD、GRAD のいずれかを指定 $\text{2nd F sin}^{-1} \text{ X } \blacktriangleright$ または X 2nd F sin^{-1} (DAns 機能)
	$\cos^{-1}x$	A C S	アークコサイン <u>arc cosine</u> (逆余弦)	$\text{2nd F cos}^{-1} \text{ X } \blacktriangleright$ または X 2nd F cos^{-1} (DAns 機能)
	$\tan^{-1}x$	A T N	アークタンジェント <u>arc tangent</u> (逆正接)	$\text{2nd F tan}^{-1} \text{ X } \blacktriangleright$ または X 2nd F tan^{-1} (DAns 機能)

(注) θ 、X については、60進数の度・分・秒の記号で指定することもできます。

項 目	通常の実現	本機の実現	読 み か た	入力方法と備考
双曲線関数	$\sinh x$	H S N	ハイパボリックサイン <u>hyperbolic sine</u>	$\text{H S N X } \blacktriangleright$ $\frac{e^x - e^{-x}}{2}$
	$\cosh x$	H C S	ハイパボリックコサイン <u>hyperbolic cosine</u>	$\text{H C S X } \blacktriangleright$ $\frac{e^x + e^{-x}}{2}$
	$\tanh x$	H T N	ハイパボリックタンジェント <u>hyperbolic tangent</u>	$\text{H T N X } \blacktriangleright$ $\frac{e^x - e^{-x}}{e^x + e^{-x}}$
逆双曲線関数	$\sinh^{-1}x$	A H S	アークハイパボリックサイン <u>arc hyperbolic sine</u>	$\text{A H S X } \blacktriangleright$ $\log_e (x + \sqrt{x^2 + 1})$ x は任意の数
	$\cosh^{-1}x$	A H C	アークハイパボリックコサイン <u>arc hyperbolic cosine</u>	$\text{A H C X } \blacktriangleright$ $\pm \log_e (x + \sqrt{x^2 - 1})$ $x \geq 1$
	$\tanh^{-1}x$	A H T	アークハイパボリックタンジェント <u>arc hyperbolic tangent</u>	$\text{A H T X } \blacktriangleright$ $\log_e \sqrt{\frac{1+x}{1-x}} \quad -1 < x < 1$

(注) X については、60進数の度・分・秒の記号で指定することもできます。

詳細については専門書参照のこと

(4)関数入力の方法 ④

項目 通常の表現	本機の表現	読みかた	入力方法と備考
60進数(度、分、秒) →10進数(度)の 変換	DEG	ディグリー to decimal degree	\Rightarrow DEG X \leftarrow または X \Rightarrow DEG (DAns 機能) ● 度、分、秒の記号については、次の“(5) 60進数の計算について”を参照してください。
10進数(度)→60進数 (度、分、秒)の変換	DMS	ディ・エム・エス to degrees, minutes, second	\leftarrow 2ndF \Rightarrow DMS X \leftarrow または X \leftarrow 2ndF \Rightarrow DMS (DAns 機能) ● 変換結果は、度、分、秒の記号で表示されます。 ● 度、分、秒の記号については、次の“(5) 60進数の計算について”を参照してください。
16進数 →10進数の変換	& H	アンド・ ヘキサデシマル & hexadecimal	\leftarrow 2ndF (& (H) h \leftarrow ただし、hは0~FFFFFFFFまでの16進数 (A~Fはアルファベットキーで入力) ● 10進数→16進数の変換については、333ページのHEX \$命令を参照してください。
直交座標 →極座標の変換 $Z = X + Yi$ → $Z = (r, \theta)$	POL	ポーラ to polar coordinates (極)	● DEG、RAD、GRADのいずれかを指定 \leftarrow 2ndF \Rightarrow r θ (X Y) \leftarrow → r の値 Z \leftarrow → θ の値 ここで、r の値を確認したいときは Y \leftarrow → r の値 θ の値を確認したいときは Z \leftarrow → θ の値
極座標→ 直交座標の変換 $Z = (r, \theta)$ → $Z = X + Yi$	REC	レクタンギュラー to rectangular coordinates (直角の、長方形)	● DEG、RAD、GRADのいずれかを指定 \leftarrow 2ndF \Rightarrow xy (r θ) \leftarrow → X の値 Z \leftarrow → Y の値 ここで、X の値を確認したいときは Y \leftarrow → X の値 Y の値を確認したいときは Z \leftarrow → Y の値

(注) 座標変換では、計算結果を変数YとZに入れますので、それまでY、Z (あるいはY \$、Z \$) に入っていた内容は消されます。

(5) 60進数 (度、分、秒) 計算について

60進数を使った度・分・秒計算ができます。

60進数の入力や表示には、数値の他に記号 (° (度)、' (分)、" (秒)) を使用します。

60進数 (度、分、秒) → 10進数 (度) の変換 (DEG)

度・分・秒の記号を使って変換するときは、VDEG命令 (363ページ) を用います。

変換結果は数値で表示されます。

〈例〉 25度45分18秒を10進数に変換

キ ー 操 作	表 示 部
\leftarrow V \leftarrow D \leftarrow E \leftarrow G \leftarrow (SHIFT) + \leftarrow 1 25 \leftarrow 2ndF \leftarrow ° 45 \leftarrow 2ndF \leftarrow ' 18 \leftarrow 2ndF \leftarrow " (SHIFT) + \leftarrow 1 \leftarrow \leftarrow	VDEG 1 2 5 ° 4 5 ' 1 8 " \leftarrow 2 5 . 7 5 5

10進数 (度) → 60進数 (度、分、秒) の変換 (DMS)

変換結果は度・分・秒の記号で表示されます。

〈例〉 123.6789度を60進数に変換

キ ー 操 作	表 示 部
123 \leftarrow 6789 \leftarrow 2ndF \Rightarrow DMS または \leftarrow 2ndF \Rightarrow DMS 123 \leftarrow 6789 \leftarrow	1 2 3 ° 4 0 ' 4 4 . 0 4 " 度 分 秒 1 2 3 ° 4 0 ' 4 4 . 0 4 " 秒の端数 (小数点以下) は、10進数になります。

(注) ● 本機は、演算結果の値と進数を記憶していますので、ラストアンサー機能を用いたときは、演算結果の内容がそのまま呼び出されます。

上記の演算例のあとでは、(ANS) を押すと) 1 2 3 ° 4 0 ' 4 4 . 0 4 " と表示されます。

● 演算結果が度・分・秒の記号で表示されたときは、 \leftarrow F \leftrightarrow E の操作は無効です。

キ ー 操 作	表 示 部
25 \leftarrow 4518 \Rightarrow DEG \leftarrow F \leftrightarrow E (指数表示に変換)	2 5 . 7 5 5 2 . 5 7 5 5 E 0 1

● 度・分・秒の記号表示が対応できる範囲は $-10^4 < X < 10^4$ です。この範囲を超えたときは、E R R O R 3 3 と表示されます。

● その他の表示例

キ ー 操 作	表 示 部
0 \leftarrow 001 \leftarrow 2ndF \Rightarrow DMS 1 \leftarrow 2ndF \Rightarrow DMS 0 \leftarrow 1 \leftarrow 2ndF \Rightarrow DMS 9999 \leftarrow 009 \leftarrow 2ndF \Rightarrow DMS	0 ° 0 0 ' 0 3 . 6 " 1 ° 0 ° 0 6 ' 9 9 9 9 ° 0 0 ' 3 2 . 4 "

度・分・秒の記号を使った演算例

- ① \leftarrow 2ndF \Rightarrow DMS 1. 2 3 + 0. 5 \leftarrow → 1. 7 3
 ② \leftarrow 2ndF \Rightarrow DEG 2 5 . 4 5 1 8 + 0. 5 \leftarrow → 2 6 . 2 5 5
 ③ VDEG 1 2 5 ° 4 5 ' 1 8 " + 0. 5 \leftarrow → 2 6 . 2 5 5

- ④ VDEG 25° 45' 18" +
 (→DEG) 25. 4518 +
 (2ndF) (→DMS) 1. 23 (↓) ⇒ 52. 74
- ⑤ 1° 20' + 2° 20' (↓) ⇒ 3° 40'
- ⑥ 3° 30' 45" + 6° 45' 36" (↓) ⇒ 10° 16' 21"
- ⑦ (→DEG) 25. 4518 + (2ndF) (→DMS) 1. 23 (↓)
 ⇒ 26. 985
- ⑧ 3° 45' - 1. 69 (↓) ⇒ 2. 06
- ⑨ 3° 30' 45" × 10 (↓) ⇒ 35° 07' 30"
- ⑩ 10 ÷ 3° 30' 45" (↓) ⇒ 2. 846975089

(注) 従来機PC-G850Sでは、60進数に変換した値をそのまま演算に使用できません。

本機では、60進数に変換した値を演算に使用できますが、同じ演算でも演算結果が異なることがあります。

(6) 取り扱う角度の単位の指定について

三角関数、逆三角関数および座標変換などの計算では、取り扱う角度の単位を正しく指定しておく必要があります。

マニュアル計算では、(2ndF) (DRG) の操作で指定できますので、度・ラディアン・グラードの指定をしたうえで計算を行ってください。なお、次の命令で指定することもできます。

角度単位	命令	表示シンボル	備考
度	DEGREE (↓)	DEG	直角を90で表す単位 [°]
ラディアン	RADIAN (↓)	RAD	直角を $\frac{\pi}{2}$ で表す単位 [rad]
グラード	GRAD (↓)	GRAD	直角を100で表す単位 [g]

これは、BASIC言語でプログラムを組むときなどに使いますので、使いなれておいてください。

2. マニュアル計算における関数計算の操作方法と練習

1 2乗 x^2

例題を始める前に (CLS) を押して表示をクリアしてください。

数字が表示されていると、関数キーを押したとき、ダイレクトアンサー機能により計算を行ってしまいます。

例題	キー操作	表示部
1 37^2	(x^2) 37 (↓) または 37 (x^2) (↓)	SQU37_ 1369. 1369.
2 $(87 \times 57)^2$	(x^2) () 87 () 57 () (↓) または 87 () 57 () (x^2) (↓)	SQU(87*57)_ 24591681. 24591681.

3 $\sqrt{5^2 - 4^2}$	($\sqrt{}$) () (x^2) 5 () (-) (x^2) 4 () (↓) または 5 (x^2) () (-) (x^2) 4 () ($\sqrt{}$) (↓)	SQR(SQU5-SQU4)_ 3. 3.
----------------------	---	--------------------------

練習問題 4

を行ってください。

2 平方根 $\sqrt{}$

例題を始める前に (CLS) を押して表示をクリアしてください。

例題	キー操作	表示部
1 $\sqrt{3}$	($\sqrt{}$) 3 (↓) または 3 ($\sqrt{}$) (↓)	SQR3_ 1.732050808 1.732050808
2 $\sqrt{25+86} \times \sqrt{37}$	($\sqrt{}$) () 25 () (+) 86 () () (*) ($\sqrt{}$) 37 (↓) または 25 () (+) 86 () ($\sqrt{}$) () (*) ($\sqrt{}$) 37 (↓)	SQR(25+86)*SQR37_ 64.08587988 64.08587986

(注) 計算のしかたにより誤差が生じます。



誤差について

本機の内部では有効数字12桁で計算し、表示するときに11桁目を四捨五入して10桁で表示します。
 計算を途中で区切ったときは、有効数字10桁にまるめた数で以後の計算を行います。
 そのため、連続計算した場合と、計算の途中で区切った場合とでは誤差が生じます。

練習問題 5

を行ってください。

3 3乗 2^{nd}F x^3

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 3^3	2^{nd}F x^3 3 ↵ または 3 2^{nd}F x^3 ↵	CUB3_ 27.
2 $2^3 + 3^3$	2^{nd}F x^3 2 $+$ 2^{nd}F x^3 3 ↵ または 2 2^{nd}F x^3 $+$ 2^{nd}F x^3 3 ↵	CUB2+CUB3_ 35.

練習問題 6

を行ってください。

4 立方根 2^{nd}F $\sqrt[3]{}$

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $\sqrt[3]{125}$	2^{nd}F $\sqrt[3]{}$ 125 ↵ または 125 2^{nd}F $\sqrt[3]{}$ ↵	CUR125_ 5.
2 $\sqrt[3]{(25+38)(96-57)}$	2^{nd}F $\sqrt[3]{}$ $($ $($ 25 $+$ 38 $)$ $*$ $($ 96 $-$ 57 $)$ $)$ ↵ または 25 $+$ 38 $)$ $*$ $($ 96 $-$ 57 $)$ $)$ 2^{nd}F $\sqrt[3]{}$ ↵	CUR((25+38)*(96-57))_ 13.49382434

練習問題 7

を行ってください。

5 逆数 $1/x$

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $\frac{1}{125}$	$1/x$ 125 ↵ または 125 $1/x$ ↵	RCP125_ 0.008
2 $\frac{1}{\sqrt{185}}$	$1/x$ $($ $\sqrt{}$ 185 $)$ ↵ または 185 $\sqrt{}$ $1/x$ ↵	RCP(SQR185)_ 0.073521462
3 $\frac{1}{\frac{1}{6} + \frac{1}{7}}$	$1/x$ $($ $1/x$ 6 $+$ $1/x$ 7 $)$ ↵ または 6 $1/x$ $-$ $1/x$ 7 $)$ $1/x$ ↵	RCP(RCP6+RCP7)_ 3.230769231

練習問題 8

を行ってください。

6 べき乗 y^x

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 8^5	8 y^x 5 ↵	$8 \wedge 5$ _ 32768.
2 $8^{5.37}$	8 y^x 5.37 ↵	$8 \wedge 5.37$ _ 70728.30171
3 5×7^4	5 $*$ 7 y^x 4 ↵	$5 * 7 \wedge 4$ _ 12005.
4 $2.7^{3.4} + 4.3^{2.5}$	2.7 y^x 3.4 $+$ 4.3 y^x 2.5 ↵	$2.7 \wedge 3.4 + 4.3 \wedge 2.5$ _ 67.62611161
5 $4^{-2} \left(\frac{1}{4^2}\right)$	4 y^x $-$ 2 ↵	$4 \wedge -2$ _ 0.0625

練習問題

9

10

11

を行ってください。

7 べき乗根 $y^x \wedge$

平方根 $\sqrt{\quad}$ ($\frac{1}{2}$ 乗)、立方根 $\sqrt[3]{\quad}$ ($\frac{1}{3}$ 乗) を一般化すると、 $\sqrt[n]{Y} = Y^{\frac{1}{n}}$ の関係になります。つまりこれは、べき乗の変形と考えられます。

べき乗根の公式

m 、 n 、 p は正の整数、 $a > 0$ 、 $b > 0$ とするとき、以下の式がなりたちます。

1. $\sqrt[n]{a} = a^{\frac{1}{n}}$
2. $(\sqrt[n]{a})^n = a$
3. $(\sqrt[n]{a})^m = \sqrt[n]{a^m} = a^{\frac{m}{n}}$
4. $\sqrt[n]{a^{\frac{m}{p}}} = a^{\frac{mp}{np}} = a^{\frac{m}{n}}$
5. $\sqrt[n]{\sqrt[p]{a}} = (\sqrt[n]{a^{\frac{1}{p}}}) = a^{\frac{1}{pn}} = \sqrt[pn]{a}$
6. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{a \cdot b}$
7. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$

練習問題

12

を行ってください。

8 階乗 $2nd F$ $n!$ 順列 nPr 組み合わせ $2nd F$ nCr

(1)階乗

n から 1 までの整数をかけ合わせて作った値を $n!$ と表し、 n の階乗と言います。

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1$$

たとえば、

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120 \text{ になります。}$$

例題を始める前に \boxed{CLS} を押して表示をクリアしてください。

例題	キー操作	表示部
1 5!	$\boxed{2nd F} \boxed{n!} 5$ $\boxed{\downarrow}$ または $5 \boxed{2nd F} \boxed{n!}$	FACT 5 _ 1 2 0 . 1 2 0 .
2 8! × 5!	$\boxed{2nd F} \boxed{n!} 8 \boxed{*} \boxed{2nd F} \boxed{n!} 5$ $\boxed{\downarrow}$	FACT 8 * FACT 5 _ 4 8 3 8 4 0 0 .

(2)順列・組み合わせ

相異なる n 個のものから r 個を取って 1 組としたものを、 n 個のものから r 個を取る 組み合わせ (Combination) といい、その組み合わせの数を ${}_nC_r$ で表します。

r 個を取り出した上で、さらに取り出した r 個のものを 1 列に順序づけて並べたものを、 n 個のものから r 個を取る 順列 (Permutation) といい、その順列の数を ${}_nP_r$ で表します。

$$\text{順列} \quad {}_nP_r = \frac{n!}{(n-r)!} \quad (r \leq n)$$

$$\text{組み合わせ} \quad {}_nC_r = \frac{n!}{r!(n-r)!} \quad \text{ここで } 0! = 1 \text{ と定義します。}$$

例題	キー操作	表示部
1 ・1 から 10 のボールから 7 つのボールを取り出して並べるしかたの数は ${}_{10}P_7 = \frac{10!}{(10-7)!}$	$\boxed{nPr} \boxed{\leftarrow} 10 \boxed{\rightarrow} 7 \boxed{\rightarrow}$ $\boxed{\downarrow}$ または $\boxed{2nd F} \boxed{n!} 10 \boxed{\div} \boxed{2nd F} \boxed{n!} \boxed{\leftarrow}$ $10 \boxed{-} 7 \boxed{\rightarrow} \boxed{\downarrow}$	NPR (1 0, 7) _ 6 0 4 8 0 0 . 6 0 4 8 0 0 .
2 ・10 チームでリーグ戦を行うときの全試合数は ${}_{10}C_2 = \frac{10!}{2!(10-2)!}$	$\boxed{2nd F} \boxed{nCr} \boxed{\leftarrow} 10 \boxed{\rightarrow} 2 \boxed{\rightarrow}$ $\boxed{\downarrow}$ または $\boxed{2nd F} \boxed{n!} 10 \boxed{\div} \boxed{\leftarrow} \boxed{2nd F} \boxed{n!}$ $2 \boxed{*} \boxed{2nd F} \boxed{n!} \boxed{\leftarrow} 10 \boxed{-} 2$ $\boxed{\rightarrow} \boxed{\rightarrow} \boxed{\downarrow}$	NCR (1 0, 2) _ 4 5 . 4 5 .

練習問題

13

14

を行ってください。

9 常用対数 \log

$a^m = M$ のとき、指数 m は a を底とする M の対数といい、 $m = \log_a M$ と表します。

このとき M を対数 m の真数と言います。

真数は常に正です。特に、底が 10 の対数を常用対数と呼び、底を略して $\log M$ と表します。

対数の基本的性質

1. $\log 1 = 0$
2. $\log 10 = 1$
3. $\log 100 = 2$
4. $\log (M \cdot N) = \log M + \log N$
(対数で表すと、掛け算がたし算になります)
5. $\log \frac{M}{N} = \log M - \log N$
(対数で表すと、わり算が引き算になります)
6. $\log M^P = P \cdot \log M$
7. $\log \sqrt[n]{M^m} = \log M^{\frac{m}{n}} = \frac{m}{n} \log M$
8. $\log_a b = \frac{\log b}{\log a}$
9. $\log_b a = \frac{1}{\log_a b}$ (底の変換公式)
10. $10^{\log M} = M$

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $\log 1000$	[log] 1000 [↵] または 1000 [log]	LOG 1 0 0 0 _ 3. 3.
2 $\log \sqrt{15}$	[log] [√] 15 [↵]	LOG SQR 1 5 _ 0. 5 8 8 0 4 5 6 2 9
3 $\log \frac{9}{15}^*$	[log] [⏏] 9 [÷] 15 [⏏] [↵]	LOG (9 / 1 5) _ - 0. 2 2 1 8 4 8 7 4 9

※ () を忘れると $\log 9 \div 15$ の計算になります。

練習問題 15

を行ってください。

10 常用指数 2nd F 10^x

10のべき乗を求めるときに用います。

これは $\log x$ の逆関数で、対数の真数を求めることになります。(58ページ逆三角関数の項参照)

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 10^3	[2nd F] [10 ^x] 3 [↵]	TEN 3 1 0 0 0 .
2 $10^{7.4} + 10^{8.3}$	[2nd F] [10 ^x] 7.4 [=] [2nd F] [10 ^x] 8.3 [↵]	TEN 7. 4 + TEN 8. 3 2 2 4 6 4 5 0 9 5. 8
3 $\log N = 1.5$ ↓ $N = 10^{1.5}$	[2nd F] [10 ^x] 1.5 [↵]	TEN 1. 5 3 1. 6 2 2 7 7 6 6

練習問題 16

を行ってください。

11 自然対数 ln

10を底とする対数が常用対数であるのに対し、

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots = 2.718281828459 \dots$$

を底とする対数を、自然対数またはネーピアの対数といいます。

自然対数と常用対数の関係

$$\log x = \frac{\ln x}{\ln 10} = 0.434294481 \times \ln x$$

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $\ln 10$	[ln] 10 [↵] または 10 [ln]	LN 1 0 _ 2. 3 0 2 5 8 5 0 9 3 2. 3 0 2 5 8 5 0 9 3
2 $\ln 18 + \ln 15$	[ln] 18 [=] [ln] 15 [↵]	LN 1 8 + LN 1 5 _ 5. 5 9 8 4 2 1 9 5 9
3 $\ln (18 \times 15)$	[ln] [⏏] 18 [*] 15 [⏏] [↵]	LN (1 8 * 1 5) _ 5. 5 9 8 4 2 1 9 5 9
4 $\ln 16 - \ln 9$	[ln] 16 [-] [ln] 9 [↵]	LN 1 6 - LN 9 _ 0. 5 7 5 3 6 4 1 4 4
5 $\ln \left(\frac{16}{9} \right)$	[ln] [⏏] 16 [÷] 9 [⏏] [↵]	LN (1 6 / 9) _ 0. 5 7 5 3 6 4 1 4 4

12 自然指数 2^{nd}F e^x

この関数は $\ln x$ の逆関数（真数）を求める関数です。（逆三角関数の項参照）

例題を始める前に CLS を押して表示をクリアしてください。

例	題	キ ー 操 作	表 示 部
1	e^1	2^{nd}F e^x 1 ↓ または 1 2^{nd}F e^x	EXP 1 _ 2.718281828
2	$e^{2.845}$	2^{nd}F e^x 2.845 ↓	EXP 2.845 _ 17.20155867
3	$e^{-6.41} \times e^{8.65}$	2^{nd}F e^x \div 6.41 \times 2^{nd}F e^x 8.65 ↓	EXP -6.41 * EXP 8.65 _ 9.393331288

2^{nd}F 10^x 、 2^{nd}F e^x 、 2^{nd}F Exp の違いについて

2^{nd}F 10^x	常用指数	10のべき乗を求めます	数式例 ① $10^{1.5}$ ② $10^{7.4}$ ③ $10^{-1.3 \times 6.4}$
2^{nd}F e^x	自然指数	eのべき乗を求めます	数式例 ① e^1 ② $e^{-2.5}$ ③ $e^3 \times e^{8.5}$
2^{nd}F Exp	指数指定	指数部を置数するときに 使います 指数は±(0~99)の整数 です。 31ページ参照	置数例 ① 2.5×10^3 2.5 2^{nd}F Exp 3 と置数 (2.5E3と表示されます) ② 2×10^{-3} 2 2^{nd}F Exp -3 と置数 (2E-3と表示されます)

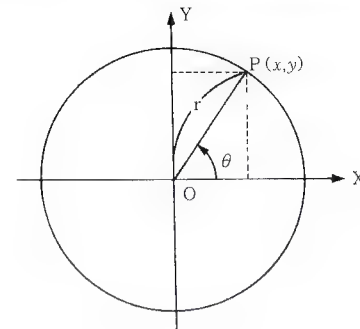
練習問題 17

を行ってください。

13 三角関数 \sin \cos \tan

正弦 ($\sin \theta$)、余弦 ($\cos \theta$)、正接 ($\tan \theta$) を求めます。

右の図のように、 xy 平面上で、動径OPが始線OX（ x 軸の正方向）となす角を θ とし、動径上の点Pの座標を (x, y) 、 $OP = \sqrt{x^2 + y^2} = r$ とすると、次のような関係がなりたちます。



$$\begin{aligned}
 1. \sin \theta &= \frac{y}{r} \quad (\text{正弦}) & \left\{ \begin{array}{l} r \\ y \end{array} \right. \sin \theta &= \frac{y}{r} \\
 2. \cos \theta &= \frac{x}{r} \quad (\text{余弦}) & \left\{ \begin{array}{l} r \\ x \end{array} \right. \cos \theta &= \frac{x}{r} \\
 3. \tan \theta &= \frac{y}{x} \quad (\text{正接}) & \left\{ \begin{array}{l} y \\ x \end{array} \right. \tan \theta &= \frac{y}{x}
 \end{aligned}$$

このように考えると
わかりやすい。

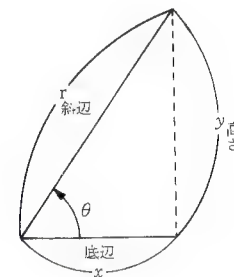
\sin は $\frac{\text{高さ}}{\text{斜辺}}$ で求められますから

Sを筆記体で書くとSとなりますので

$\frac{\text{斜辺}}{\text{高さ}}$ で $\frac{y}{r}$ になるという意味です。

Cosの場合もCを∠と考えて $\frac{\text{斜辺}}{\text{底辺}}$ で $\frac{x}{r}$

Tanの場合もTを「」と考えて $\frac{\text{高さ}}{\text{底辺}}$ で $\frac{y}{x}$



■角度の単位

半径に等しい長さの弧に対する中心角は、円の大きさに関係なく一定です。

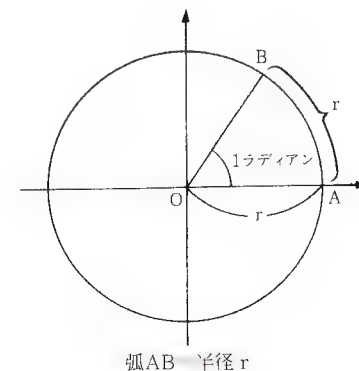
この一定の角を単位1（ラジアン）として、角を測る方法が弧度法です。

弧度法では、単位の名称を略して、角の大きさを無名数で表すのが普通です。

弧度法に対して、度を単位として角を測る方法（1直角 = 90° 、 $1^\circ = 60'$ 、 $1' = 60''$ ）を60分法といいます。

また、この 90° を 100° （グラード）として測る方法もあります。

60分法	弧度法	グラード法
ディグリー	ラジアン	グラード
DEG	RAD	GRAD
$90^\circ \longleftrightarrow$	$\pi/2 \longleftrightarrow$	$100^\circ \longleftrightarrow$
$180^\circ \longleftrightarrow$	$\pi \longleftrightarrow$	$200^\circ \longleftrightarrow$
$360^\circ \longleftrightarrow$	$2\pi \longleftrightarrow$	$400^\circ \longleftrightarrow$



弧AB 半径 r

(2)三角関数の計算を行う前に

三角関数の計算を行うときは、角度単位の指定をしてください。

まず、与えられた計算式において

1. 60分法で求めるのか
2. 弧度法で求めるのか
3. グラードで求めるのか

これをまず確認してください。そして、

- ① 60分法（ディグリー）で求めるなら、表示部の右側に小さな英字でDEGが出ていることを確認してください。もしDEGが出ていないならDEGという英字が出るまで **[2nd F] [DRG]** を繰り返してください。（DEG表示は、電源ON時に必ず表示されます。）
- ② 弧度法（ラジアン）で求めるなら、表示部にRADが出るまで **[2nd F] [DRG]** を繰り返してください。
- ③ グラード法（グラード）で求めるなら、表示部にGRADが出るまで **[2nd F] [DRG]** を繰り返してください。



GRAD（グラード）について

GRADはヨーロッパの測量関係など特殊な業務で使用されている角度です。

日本ではあまりなじみがありませんが、直角が100°と考えると数値のため、計算が楽になります。

■主な角の三角関数

	60分法	0°	30°	45°	60°	90°	120°	135°	150°	180°	270°	360°
三角関数	弧度法	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	$\frac{2\pi}{3}$	$\frac{3\pi}{4}$	$\frac{5\pi}{6}$	π	$\frac{3\pi}{2}$	2π
$\sin \theta$		0	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{\sqrt{3}}{2}$	1	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	0	-1	0
$\cos \theta$		1	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	$-\frac{1}{\sqrt{2}}$	$-\frac{\sqrt{3}}{2}$	-1	0	1
$\tan \theta$		0	$\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$	∞	$-\sqrt{3}$	-1	$-\frac{1}{\sqrt{3}}$	0	∞	0
図による 角度と符号	<div style="display: flex; flex-direction: column; align-items: flex-end;"> <div>sin の 符号</div> <div>cos の 符号</div> <div>tan の 符号</div> </div>											

① SIN関数

例題を始める前に **[CLS]** を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $\sin 60^\circ$	表示部にDEGを表示させます [sin] 60 [↵] または 60 [sin]	SIN 60 0.866025403 0.866025403
2 $\sin \frac{\pi}{3}$	表示部にRADを表示させます [sin] [⏏] [π] [÷] 3 [⏏] [↵] または [π] [÷] 3 [↵] [sin]	SIN(PI/3) 0.866025403 0.866025403
3 $\sin (0.5\pi + 2.4)$	表示部にRADを表示させます [sin] [⏏] 0.5 [*] [π] [+] 2.4 [⏏] [↵]	SIN(0.5*PI+2.4) -0.737393715
4 $\sin^2 30^\circ$	表示部にDEGを表示させます [sin] 30 [y^x] 2 [↵] または 30 [sin] [y^x] 2 [↵]	SIN 30 ^ 2 0.25 0.25
5 $\sin 18^\circ 30'$	表示部にDEGを表示させます [sin] [→DEG] 18.30 [↵] または [→DEG] 18.30 [↵] [sin] または 18.30 [→DEG] [sin]	SIN DEG 18.30 0.317304656 0.317304656 0.317304656
6 $\sin 250^\circ$	表示部にGRADを表示させます [sin] 250 [↵] または 250 [sin]	SIN 250 -0.707106781 -0.707106781
7 $\sin 25^\circ 45' 18''$	表示部にDEGを表示させます [sin] 25 [2nd F] [°] 45 [2nd F] ['] 18 [2nd F] ["] [↵] または 25 [2nd F] [°] 45 [2nd F] ['] 18 [2nd F] ["] [sin]	SIN 25° 45' 18" 0.434523856 0.434523856



角度指定について

三角関数の計算を行う場合は、DEG、RAD、GRADの角度指定が必要です。角度指定をまちがうと正しい結果が得られません。角度指定は **[2nd F] [DRG]** を押して切り替えます。

2 COS関数

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $4 \times \cos 25^\circ$	表示部にDEGを表示させます 4 [*] [cos] 25 [↵]	4 * COS 25 _ 3.625231148
2 $\cos \frac{\pi}{5}$	表示部にRADを表示させます [cos] [⏏] [π] [÷] 5 [⏏] [↵]	COS (PI / 5) _ 0.809016994
3 $\cos(473^\circ + 168^\circ)$	表示部にGRADを表示させます [cos] [⏏] 473 [⏏] 168 [⏏] [↵]	COS (473 + 168) _ -0.799684658
4 $\cos 25^\circ 45' 18''$	表示部にDEGを表示させます [cos] 25 [2ndF] [°] 45 [2ndF] ['] 18 [2ndF] ["] [↵] または 25 [2ndF] [°] 45 [2ndF] ['] 18 [2ndF] ["] [cos]	COS 25° 45' 18" _ 0.900660323 0.900660323

3 TAN関数

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $\tan 19.7^\circ$	表示部にDEGを表示させます [tan] 19.7 [↵]	TAN 19.7 _ 0.358051837
2 $\tan (24^\circ 8' 55'' + 37^\circ 19' 23'')$	表示部にDEGを表示させます [tan] [⏏] [→DEG] 24.0855 [⏏] [→DEG] 37.1923 [⏏] [↵]	TAN (DEG 24.0855 + DEG 37.1923) _ 1.839600915
3 $\tan (\pi + \frac{3}{8}\pi)$	表示部にRADを表示させます [tan] [⏏] [π] [+] 3 [⏏] 8 [*] [π] [⏏] [↵]	TAN (PI + 3 / 8 * PI) _ 2.414213562
4 $\tan 25^\circ 45' 18''$	表示部にDEGを表示させます [tan] 25 [2ndF] [°] 45 [2ndF] ['] 18 [2ndF] ["] [↵] または 25 [2ndF] [°] 45 [2ndF] ['] 18 [2ndF] ["] [tan]	TAN 25° 45' 18" _ 0.482450314 0.482450314

練習問題 18

を行ってください。

14 逆三角関数 [2ndF] [sin⁻¹]、[2ndF] [cos⁻¹]、[2ndF] [tan⁻¹]

逆三角関数とは、三角関数 sin、cos、tan のそれぞれの逆関数です。

逆関数とは、 $y = x$ という一次関数に対して対称になる関数をいいます。

常用対数	↔	常用指数	それぞれ $x = y$ に対して対称な関数 となります
自然対数	↔	自然指数	
これと同じように、	↔	逆三角関数	
三角関数	↔	逆三角関数	

$$y = \sin x \longleftrightarrow x = \sin^{-1} y \quad (\text{アークサインワイと読む})$$

$$y = \cos x \longleftrightarrow x = \cos^{-1} y \quad (\text{アークコサインワイと読む})$$

$$y = \tan x \longleftrightarrow x = \tan^{-1} y \quad (\text{アークタンジェントワイと読む})$$

逆三角関数の計算は、三角関数と同様に DEG、RAD、GRAD の指定が必要です。

今、上の式の x を θ とおけば ($x = \theta$)

三 角 関 数	逆三角関数	逆三角関数の値の範囲		
		D E G	R A D	G R A D
$y = \sin \theta$	$\theta = \sin^{-1} y$	$-90^\circ \leq \theta \leq 90^\circ$	$-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$	$-100^\circ \leq \theta \leq 100^\circ$
$y = \cos \theta$	$\theta = \cos^{-1} y$	$0^\circ \leq \theta \leq 180^\circ$	$0 \leq \theta \leq \pi$	$0^\circ \leq \theta \leq 200^\circ$
$y = \tan \theta$	$\theta = \tan^{-1} y$	$-90^\circ \leq \theta \leq 90^\circ$	$-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$	$-100^\circ \leq \theta \leq 100^\circ$

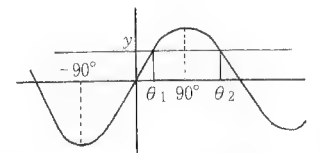
- 三角関数 $\sin \theta$ 、 $\cos \theta$ の値は、すべて -1 から $+1$ の値にありますから、それ以外の値で $\sin^{-1} y$ 、 $\cos^{-1} y$ の計算はできません。



逆三角関数の値の範囲について

$\sin \theta$ の関数は右図のようになります。

$\sin^{-1} y$ は y のときの θ を求めますが、右図の y の場合、計算機では求められる θ が1つしか存在しない $-90^\circ \sim 90^\circ$ の範囲に限定しています。 θ_2 を知りたいときは、 $180^\circ - \theta_1$ を計算してください。

1 sin⁻¹、cos⁻¹、tan⁻¹

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 $\sin^{-1}0.5$ 角度単位 (DEG)	表示部にDEGを表示させます $(2nd F) (\sin^{-1}) 0.5$ または $0.5 (2nd F) (\sin^{-1})$	ASN0.5_ 30. 30.
2 $\cos^{-1}0.628$ (RAD)	表示部にRADを表示させます $(2nd F) (\cos^{-1}) 0.628$ または $0.628 (2nd F) (\cos^{-1})$	ACS0.628_ 0.891815777 0.891815777
3 $\tan^{-1}1$ (GRAD)	表示部にGRADを表示させます $(2nd F) (\tan^{-1}) 1$ または $1 (2nd F) (\tan^{-1})$	ATN1_ 50. 50.
4 $2 \sin^{-1}0.785$ (DEG)	表示部にDEGを表示させます $2 (*) (2nd F) (\sin^{-1}) 0.785$ 度分秒で求める場合は、ここで $(2nd F) (\rightarrow DMS)$	2 * ASN0.785_ 103.4413565 (103,4413565度) 103° 26' 28.88" (103度26分28.88秒)
5 $\cos^{-1}0.43 + \cos^{-1}0.66$ (RAD)	表示部にRADを表示させます $(2nd F) (\cos^{-1}) .43 (+) (2nd F) (\cos^{-1}) .66$	ACS.43+ACS.66_ 1.976281116
6 $\tan^{-1} \sqrt{\frac{1-0.6^2}{0.6}}$ (DEG)	表示部にDEGを表示させます $(2nd F) (\tan^{-1}) (\sqrt{}) (1 - 0.6^2) (/) 0.6$	ATNSQR((1-.6^2)/.6)_ 45.92428582

練習問題 19

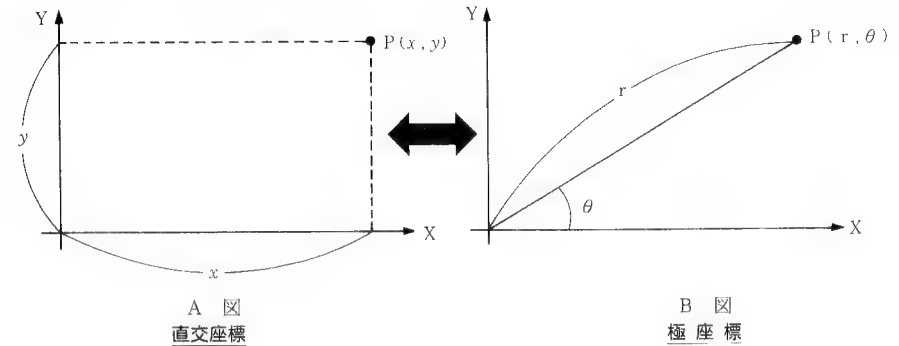
を行ってください。

15 座標変換

座標変換とは、ベクトルや複素数において、 x 成分・ y 成分で表される式を大きさ r ・偏角 θ に変換することや、その逆に、大きさ r ・偏角 θ から x 成分・ y 成分に変換することをいいます。

x 成分、 y 成分として表されるものを、**直交座標** $Z = x + yi$
 大きさ r 、偏角 θ として表されるものを **極座標** $Z = (r, \theta)$
 と、いいます。

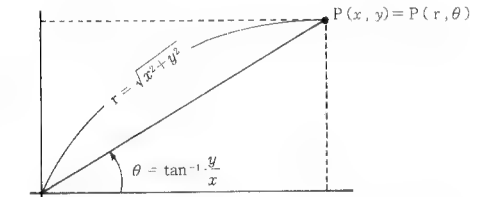
この座標変換の場合も、三角関数同様、角度を扱いますので、角度指定DEG、RAD、GRADのいずれかを指定する必要があります。



A図も、B図もXY平面上の点Pを表現する方法です。表現方法は違いますが、点Pの位置を示していることにおいては同じ結果となります。

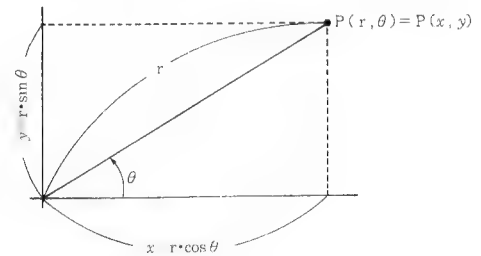
直交座標を極座標に変換するには

- 極座標の r を求める $\longrightarrow r = \sqrt{x^2 + y^2}$
- 偏角 θ を求める $\longrightarrow \theta = \tan^{-1} \frac{y}{x}$



極座標を直交座標に変換するには

- 直交座標の x を求める $\longrightarrow x = r \cos \theta$
- 直交座標の y を求める $\longrightarrow y = r \sin \theta$



1 直交座標 → 極座標変換 [2nd F] → $r\theta$

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 (3,8)を極座標に変換 ($P = 3 + 8i$)	角度単位を“度”に指定します 表示部に DEG を表示させます [2nd F] → $r\theta$ [C] 3 [.] 8 [D] [↵] [Z] [↵]	POL(3, 8) 8.544003745 (rの値) Z_ 69.44395478 (θの値)
	[解説] 直交座標 (3, 8) は極座標に変換すると約 (8.54, 69.4°) です ここで r の値を確認したいときは [Y] [↵] これは r の値が変数 Y に収められていることを意味します 次に θ の値を確認したいときは [Z] [↵] θ の値は変数 Z に収められていることを意味します さらに θ の値は度で表されているので度・分・秒に変換したいときは [2nd F] → DMS	Y_ 8.544003745 Z_ 69.44395478 69° 26' 38.24" (69度26分38.24秒)
2 (3, -8)を極座標に変換 ($P = 3 - 8i$)	表示部に DEG を表示させます [2nd F] → $r\theta$ [C] 3 [.] [-] 8 [D] [↵] [Z] [↵]	POL(3, -8) 8.544003745 (rの値) Z_ -69.44395478 (θの値)
	[解説] 直交座標 (3, -8) は極座標に変換すると約 (8.54, -69.4°) です	
3 (-3, -8)を極座標に変換 ($P = -3 - 8i$)	表示部に DEG を表示させます [2nd F] → $r\theta$ [C] [-] 3 [.] [-] 8 [D] [↵] [Z] [↵]	POL(-3, -8) 8.544003745 (rの値) Z_ -110.5560452 (θの値)

2 極座標 → 直交座標変換 [2nd F] → xy

例題を始める前に [CLS] を押して表示をクリアしてください。

例 題	キ ー 操 作	表 示 部
1 (2, 30°)を直交座標に変換 $P = (2, 30^\circ)$	表示部に DEG を表示させます [2nd F] → xy [C] 2 [.] 30 [D] [↵] [Z] [↵]	REC(2, 30) 1.732050808 (xの値) Z_ 1. (yの値)
	[解説] 極座標 (2, 30°) は直交座標に変換すると約 (1.73, 1) です ● x の値を再確認するときは [Y] [↵] 1.732050808 x の値は変数 Y に収められていることを意味します ● y の値を再確認するときは [Z] [↵] 1. y の値は変数 Z に収められていることを意味します	
2 (2, -30°)を直交座標に変換 $P = (2, -30^\circ)$	表示部に DEG を表示させます [2nd F] → xy [C] 2 [.] [-] 30 [D] [↵] [Z] [↵]	REC(2, -30) 1.732050808 Z_ -1.
3 $(2, \frac{\pi}{3})$ を直交座標に変換 $P = (2, \frac{\pi}{3})$	表示部に RAD を表示させます [2nd F] → xy [C] 2 [.] [π] [D] 3 [D] [↵] [Z] [↵]	REC(2, $\pi/3$) 1. Z_ 1.732050808

練習問題 20

を行ってください。

16 統計計算

たくさんのデータをひとつのまとまった資料にするためには、統計的な処理が必要です。

本機は統計モードで、一変数統計計算、二変数統計計算ができます。

一変数統計計算は、たとえばテストの点数のような 1 種類のデータを用いて、その平均値、標準偏差などの統計量を求めることができます。

二変数統計計算は、たとえば身長と体重のように関連がある (と予想される) 2 種類のデータを用いて、それぞれのデータの平均や標準偏差を求めたり、2 種類のデータの相関関係を調べたり、また一次回帰線による推定を行うことができます。

(1)統計モードの設定と解除の方法

1. 統計モードにするときは

RUN（またはPRO）モードで

(2nd F) (STAT)

と押します。統計モードになり、右の一変数／二変数選択画面が表示されます。

なお、統計モードでは、画面右側に“STAT”が点灯します。

この選択画面のとき、**(1)**で一変数統計計算、**(2)**で二変数統計計算が選べます。

```

***** トウケイ フンセキ *****
1: 1ヘンスウ トウケイ (x)          STAT
2: 2ヘンスウ トウケイ (x, y)       CAPS
                                     DEG
ハコウ ウヲ エラント クタサイ.
```

2. 統計モードを解除するときは

(2nd F) (STAT)

と押します。RUNモードになります。

(2)一変数統計計算

1. 一変数統計計算で求める統計量

一変数統計計算では、次の統計量を求めることができます。

- サンプル数 n
- サンプルの総和 Σx

• サンプルの標準偏差 $s = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n-1}}$

母集団より抽出されたサンプルデータから、母集団の標準偏差を推定する場合に使用します。

- サンプルの2乗の和 Σx^2
- 平均 $\bar{x} = \frac{\Sigma x}{n}$

• 母標準偏差 $\sigma = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n}}$

母集団のすべてをサンプルデータとして、その標準偏差を求める場合、またはサンプルを母集団とみなして、その標準偏差を求める場合に使用します。

2. 一変数統計計算の選択

(2nd F) (STAT) と押して統計モードにした後、画面に従って **(1)** を押せば一変数統計計算が選択され、処理選択画面になります。

(2nd F) (STAT)

```

***** トウケイ フンセキ *****
1: 1ヘンスウ トウケイ (x)
2: 2ヘンスウ トウケイ (x, y)
ハコウ ウヲ エラント クタサイ.

*** ショリ *** (x)
1: ニュウリョク 2: サクシヨ/クリア
3: フンセキ 4: フリント
ハコウ ウヲ エラント クタサイ.
```

(1)

処理選択画面では、それぞれ次の処理を選ぶことができます。

- (1)** ……入力 データの入力を行うときに選びます。
- (2)** ……削除／クリア まちがったデータを入力した場合などに、そのデータを削除するときや、

新たに統計計算を始める場合に、前に計算した統計量をすべて消去するときを選びます。

(3) ……分析

統計量を求めるときに選びます。

(4) ……プリンタ

統計量を印字するときを選びます。ただし、プリンタが接続されているときにのみ選ぶことができます。プリンタが接続されていないときは **(4)** を押しても何も変わりません。

なお、一つ前の選択画面に戻すときは **BREAK (ON)** を押します。

3. データの入力方法

処理選択画面で **(1)** を押せばデータ入力画面になり、データが入力できます。

(1)

```

** データ ニュウリョク **
1: x
データが入ります。
何個目のデータかを示します。
```

①データを1つずつ入力する場合は

データ **(1)**

と押します。

②負のデータを入力する場合は

(-) データ **(1)**

と押します。

③同じデータが複数個ある場合は

データ **(1)** 個数 **(1)**

と押します。

データをすべて入力したら **BREAK (ON)** を押します。処理選択画面に戻ります。

[補足]

統計では、同じデータが何個ある場合は“度数”という言葉を使います。たとえば、同じデータが3個ある場合は“度数3”のように表します。

4. 統計量を求める方法

処理選択画面で **(3)** を押せば、次の分析画面になります。

(3)

```

** フンセキ ** (x)
1: n 2: Σx 3: Σx² 4: x
5: s 6: σ
ハコウ ウヲ エラント クタサイ.
```

(1) ~ (6) で次の統計量を求めることができます。

(1) …… n サンプル数

(2) …… Σx サンプルの総和

(3) …… Σx^2 サンプルの2乗の和

(4) …… \bar{x} 平均値

(5) …… s サンプルの標準偏差

(6) …… σ 母標準偏差

処理選択画面に戻すときは、**BREAK (ON)** を押します。

5. 新たに統計計算を行うときは（前の計算内容を消すときは）

次の2つの方法があります。

①いったん統計モードを解除し、改めて統計モードにすれば消去されます。（統計モードを解除したとき、統計量の一部は変数に保存されています。くわしくは69ページを参照してください。）

②削除／クリア機能で消去します。まず、処理選択画面にして次のようにキーを押します。

②

削除／クリア選択画面になります。

*** サクシ ョ / クリア ***

1 : データ サクシ ョ
2 : オール クリア

ハ ン コ ー ヲ エ ラ ン テ ー ク タ ー サ イ .

②

消去確認画面になります。

* オール クリア *

1 : YES
2 : NO

ハ ン コ ー ヲ エ ラ ン テ ー ク タ ー サ イ .

① 前の計算内容が消去され、処理選択画面に戻ります。

②を押した場合は、計算内容を消去せずに処理選択画面に戻ります。

■例題

ある試験の点数を、ランダムに選出した35人について見た場合、右のようになりました。

これより平均値、標準偏差を求めなさい。

No.	点数	人数	No.	点数	人数
1	30	1	5	70	8
2	40	1	6	80	9
3	50	4	7	90	5
4	60	5	8	100	2

キ ー 操 作	表 示 部
<ul style="list-style-type: none"> 統計モードにします。 (2nd F) (STAT) 一変数統計計算を選びます。 (1) “入力”を選びます。 (1) データを入力します。 30 (←) 40 (←) 50 (←) 4 (←) 60 (←) 5 (←) 70 (←) 8 (←) 80 (←) 9 (←) 90 (←) 5 (←) 100 (←) 2 (←) これでデータの入力は終わりです。 処理選択画面に戻ります。 BREAK (ON) “分析”を選びます。 (3) 平均値を求めます。 (4) サンプルの標準偏差を求めます。 (5) 母標準偏差を求めます。 (6) 処理選択画面に戻ります。 BREAK (ON) 	<p>3 4 : x = 1 0 0 . , 2 .</p> <p>3 6 : x =</p> <p>x = 7 1 . 4 2 8 5 7 1 4 3</p> <p>s = 1 6 . 4 7 5 0 8 9 4 2</p> <p>σ = 1 6 . 2 3 8 0 2 5 4 2</p>

[補足]

サンプル数、総和、2乗の和を求めるときは、分析画面でそれぞれ①、②、③を押します。

途中結果として平均値や標準偏差などの統計量を求めた後、もう一度、処理選択画面で“入力”を選んでデータを入力すれば、続きのデータとして入力できます。

6. データの削除

データ入力でまちがったデータを入れた場合などに使用する機能です。

処理選択画面にして次のようにキーを押すとデータ削除画面になります。

②

*** サクシ ョ / クリア ***

1 : データ サクシ ョ
2 : オール クリア

ハ ン コ ー ヲ エ ラ ン テ ー ク タ ー サ イ .

①

* データ サクシ ョ *
x = _

この画面で、データ入力のととき同様の操作で、まちがったデータや削除したいデータを入力すれば削除されます。データ入力のととき同様に (BREAK) (ON) を使って複数個のデータを削除することもできます。削除した後、(ON) (1) と押してデータ入力画面にしてから正しいデータを入力すれば、続きのデータとして入力できます。

[補足]

入力していない数値を削除したいデータとして入力することもできます。つまり、30と40の入力に対し、削除したいデータとして35を入力することも可能ですが、正しい結果は得られません。データを削除するときはご注意ください。

7. 統計量の印字

CE-126P（プリンタ）をお持ちの場合は、データを入力した後、計算された統計量を印字できます。プリンタを本機に接続して電源を入れた後、データを入力します。次に処理選択画面で、(4)を押して“プリンタ”を選べば印字が開始されます。

*** ショリ *** (x)

1 : ニュウリョク 2 : サクシ ョ クリア
3 : プリンタ 4 : プリンタ

ハ ン コ ー ヲ エ ラ ン テ ー ク タ ー サ イ .

④

*** インン チュー ***

印字が終われば処理選択画面に戻ります。

〈印字例〉65ページの例題のデータが入っている場合

```

n=          35.
Σx=         2500.
Σx²=        187800.
MEAN(x)=    71.42857143
s=          16.47508942
s̄=          16.23802542

```

(3) 二変数統計計算

二変数統計計算の基本的な操作方法是、一変数統計計算と同じです。先に一変数統計計算の説明をお読みください。

1. 二変数統計計算で求める統計量

二変数統計計算では、次の統計量を求めることができます。

- $n, \Sigma x, \Sigma x^2, x$ は一変数統計計算と同じ。 $sx, \sigma x$ は一変数統計計算の s, σ と同じ。
- Σy サンプル (y) の総和
- Σy^2 サンプル (y) の2乗の和
- Σxy サンプル (x, y) の積の和
- $\bar{y} \quad \bar{y} = \frac{\Sigma y}{n}$ サンプル (y) の平均値
- $s_y \quad s_y = \sqrt{\frac{\Sigma y^2 - n\bar{y}^2}{n-1}}$ サンプル (y) から求める、母数を (n-1) としたときの標準偏差
- $\sigma_y \quad \sigma_y = \sqrt{\frac{\Sigma y^2 - n\bar{y}^2}{n}}$ サンプル (y) から求める、母数を (n) としたときの標準偏差
- $a \quad a = \bar{y} - b \bar{x}$ 一次回帰線 $y = a + b x$ の係数
- $b \quad b = \frac{S_{xy}}{S_{xx}}$ 一次回帰線 $y = a + b x$ の係数
- $r \quad r = \frac{S_{xy}}{\sqrt{S_{xx} \cdot S_{yy}}}$ 相関係数
- $x' \quad x' = \frac{y - a}{b}$ 推定値 (y の値から x の値を推定する)
- $y' \quad y' = a + b x$ 推定値 (x の値から y の値を推定する)

[補足]

$$S_{xx} = \Sigma x^2 - \frac{(\Sigma x)^2}{n} \quad S_{yy} = \Sigma y^2 - \frac{(\Sigma y)^2}{n} \quad S_{xy} = \Sigma xy - \frac{\Sigma x \cdot \Sigma y}{n}$$

2. 二変数統計計算の選択

[2nd F] (STAT) と押して統計モードにした後、画面に従って **[2]** を押せば二変数統計計算が選択されて、処理選択画面になります。

3. データの入力方法

処理選択画面で **[1]** を押せばデータ入力画面になります。画面に従って、x, y のデータを入力してください。

① データが1個 (1組) の場合

データ x **[←]** データ y **[←]**

と押します。

② 同じデータが複数個ある場合

データ x **[←]** データ y **[←]** 個数 **[←]**

と押します。

③ 負数のデータは、それぞれのデータの前に **[−]** を押します。

データをすべて入力したら **BREAK** **[ON]** を押します。処理選択画面に戻ります。

4. 統計量を求める方法

処理選択画面で **[3]** を押せば分析画面になります。二変数統計計算では分析画面が2画面あります。

画面の切り替えは **[▼]** **[▲]** で行います。

処理選択画面で **[3]** を押した場合

(“分析” の第1画面です)

```

** フォンセキ **      (x, y)
1: n      2: Σx      3: Σx²      4: x̄
5: sx      6: σx      7: Σy      8: Σy²
ハコウ ウヲ エランダ クタサイ.

```

[▼]

(“分析” の第2画面です)

```

** フォンセキ **      (x, y)
1: Σxy      2: ȳ      3: sy      4: σy
5: a      6: b      7: r      8: x̄      9: ȳ
ハコウ ウヲ エランダ クタサイ.

```

[▲] で前の (第1) 画面に戻ります。

■例題

次の表は、ある地方の山桜の開花日 (4月) と同地3月の平均気温の表です。

これより、一次回帰線 $y = a + b x$ の係数 a, b と相関係数 r を求め、3月の平均気温が9.1度の場合の開花日および4月10日に開花した年の3月の平均気温を推定します。

年	1	2	3	4	5	6	7	8
平均気温 (x度)	6.2	7.0	6.8	8.7	7.9	6.5	6.1	8.2
開花日 (y日)	13	9	11	5	7	12	15	7

キ ー 操 作	表 示 部
<ul style="list-style-type: none"> 統計モードにします。 $\text{[2nd F]} \text{[STAT]}$ 二変数統計計算を選びます。 [2] “入力”を選びます。 [1] データを入力します。 <ul style="list-style-type: none"> 6.2 $\text{[.]} \text{[6]} \text{[.]}$ 13 $\text{[.]} \text{[1]} \text{[3]}$ 7.0 $\text{[.]} \text{[7]} \text{[.]}$ 9 $\text{[.]} \text{[9]}$ 6.8 $\text{[.]} \text{[6]} \text{[.]}$ 11 $\text{[.]} \text{[1]} \text{[1]}$ 8.7 $\text{[.]} \text{[8]} \text{[.]}$ 5 $\text{[.]} \text{[5]}$ 7.9 $\text{[.]} \text{[7]} \text{[.]}$ 7 $\text{[.]} \text{[7]}$ 6.5 $\text{[.]} \text{[6]} \text{[.]}$ 12 $\text{[.]} \text{[1]} \text{[2]}$ 6.1 $\text{[.]} \text{[6]} \text{[.]}$ 15 $\text{[.]} \text{[1]} \text{[5]}$ 8.2 $\text{[.]} \text{[8]} \text{[.]}$ 7 $\text{[.]} \text{[7]}$ 	$8 : x = 8.2$ $y = 7.$ $9 : x = _$
これでデータの入力は終わりです。 ・処理選択画面に戻します。 BREAK [ON] ・“分析”を選びます。 [3] ・第2画面を呼び出します。 [V] ・係数 a を求めます。 [5] ・係数 b を求めます。 [6] ・相関係数 r を求めます。 [7] ・開花日を推定します。 [9] 平均気温を入力 9.1 $\text{[.]} \text{[9]} \text{[1]}$ $a = 34.44951017$ $b = -3.425018839$ $r = -9.691068372E-01$ $x = _ (-0.9691068372)$ $y = 3.281838734$ (推定：4月3日ごろ開花)	
・“分析”の第2画面に戻します。 BREAK [ON] ・平均気温を推定します。 [8] 開花日を入力 10 $\text{[.]} \text{[1]} \text{[0]}$ $y = _$ $x = 7.13850385$ (推定：3月の平均気温は約7.1℃)	
統計モードを解除 $\text{[2nd F]} \text{[STAT]}$	

[補足]

統計計算の統計量のうち、次のものは変数U～Zに入れられ、統計モードを解除しても保持されています。したがって、RUNモードでも、この統計量を使って計算ができます。

変 数	U	V	W	X	Y	Z
統計量						
一変数統計	—	—	—	Σx^2	Σx	n
二変数統計	Σy^2	Σy	Σxy	Σx^3	Σx	n

なお、この内容は統計モードになったときに消去されます。

練習問題 21

を行ってください。

第3章

算術代入計算

1. 算術代入計算

今までは、数値と数値の計算のやりかたなどを練習してきましたが、複雑な数式や数値のたくさんあるような式を計算処理していくときに、どうしても数式（数値を代入するので）が長くなってしまいます。こんなとき、あらかじめ数値を変数キー（アルファベットキー）に記憶させておき、変数と変数の計算によって答えを求めることができます。

変数として使える文字

変数として使える文字は、 [英文字] 、 [英文字] [英文字] の組み合わせ、 [英文字] [数字] の組み合わせの3種類で、最高2文字分の長さまで使えます。

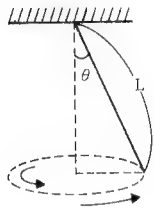
種 類	例
1. [英文字] (アルファベット)	$\text{[A]} \sim \text{[Z]}$ $A = 1$ $B = 100$ $Z = 2.5$
2. [英文字] [英文字]	$\text{[A]} \sim \text{[Z]}$ $A A = 1.5$ $M M = 30.5$ $L X = 0.003$
3. [英文字] [数字]	$\text{[A]} \sim \text{[Z]}$ $\text{[0]} \sim \text{[9]}$ $A 1 = 6.28$ $G 1 = 9.8$ $T 2 = 358.2$ $X 1 = 3$ $Y 2 = 6$

(注) ● アルファベットの小文字は大文字と同じ扱いになります。変数としてアルファベットの小文字を使用しても大文字に変換されます。

● 変数U、V、W、X、Y、Zは座標変換や統計計算で使用するため、変数として使うときには注意が必要です。

2. 例題と解説

■例題①



左図のような円すい振り子の周期 T は、糸の長さを L 、糸の鉛直となす角を θ とすると、

$$T = 2\pi \sqrt{\frac{L \cos \theta}{g}} \quad g = 9.8 \text{ [m/sec}^2\text{]}$$

で求めることができます。 $L = 50 \text{ cm}$ $\theta = 25^\circ$ の場合の周期 T を求めなさい。

このような例題をプログラム化しますと、141ページの例題⑧のようなものが考えられます。

■解説①

計算の内容	キ ー 操 作	表 示 部
$g = 9.8$ Gに9.8を代入します	$\boxed{G} = 9.8$	9.8
$L = 0.5$ Lに0.5を代入します	$\boxed{L} = 0.5$	0.5
$\theta = 25^\circ$ θ がありませんのでSに25を代入します	$\boxed{S} = 25$	25.
$T = 2\pi \sqrt{\frac{L \cos \theta}{g}}$ 文字式として入力します	$\boxed{T} = 2 * \boxed{\pi} * \boxed{\sqrt{\frac{L * \boxed{\cos} S}{G}}}$	$T = 2 * \pi * \text{SQR}(L * \cos S / G)$ 1.351106827 答 1.35秒
計算の結果はTに入っています と押せば呼び出されます	\boxed{T}	1.351106827



プレイバックについて

$\boxed{\rightarrow}$ を押して計算を行った後に $\boxed{\rightarrow}$ または $\boxed{\leftarrow}$ を押すと、計算した式が呼び戻され、再度計算ができます。エラーになったときは、エラーになった位置にカーソルが表示されます。訂正して再び $\boxed{\rightarrow}$ を押すと答が得られます。なお、カーソルは $\boxed{\rightarrow}$ を押したときは式の先頭に、 $\boxed{\leftarrow}$ を押したときは式の最後に表示されます。

■例題②

月の軌道を地球を中心とする円とみなし、地球の半径 r 、月の周期を T とすると、月の軌道半径は

$$R = \sqrt[3]{\frac{g \cdot r^2 \cdot T^2}{4\pi^2}}$$

で与えられます。地球の半径 $r = 6.4 \times 10^6 \text{ [m]}$ 、月の周期 $T = 27 \text{ 日 } 8 \text{ 時間}$ としたときの月の軌道半径を求めなさい。

27日8時間は $(27 \times 24 + 8) \times 60 \times 60$ で秒に換算され、 $T = 2.3616 \times 10^6 \text{ [sec]}$ になります。

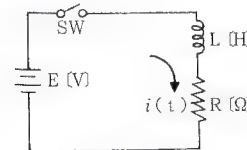
また、 g は例題①のように $g = 9.8 \text{ [m/sec}^2\text{]}$ とします。

■解説②

計算の内容	キ ー 操 作	表 示 部
$R = 6.4 \times 10^6$	$\boxed{R} = 6.4 \boxed{2nd} \boxed{F} \boxed{Exp} 6$	$R = 6.4 E 6$ 6 4 0 0 0 0 0.
$T = 2.3616 \times 10^6$	$\boxed{T} = 2.3616 \boxed{2nd} \boxed{F} \boxed{Exp} 6$	$T = 2.3616 E 6$ 2 3 6 1 6 0 0.
$G = 9.8$	$\boxed{G} = 9.8$	9.8
$RR = \sqrt[3]{\frac{G \cdot R^2 \cdot T^2}{4\pi^2}}$	$\boxed{RR} = \boxed{2nd} \boxed{\sqrt[3]{\frac{G * \boxed{R} \boxed{Y} \boxed{2} * \boxed{T} \boxed{Y} \boxed{2}}{4 * \boxed{\pi} \boxed{2}}}}$	$RR = \text{CUR}(G * (R \wedge 2) * (T \wedge 2) / (4 * (\pi \wedge 2)))$ 3 8 4 1 9 0 2 3 5.2 (m)
ここで R は地球の半径で使っているので月の軌道半径の変数を RR とします		答 384190km

このように、原式の文字に対する数値を記憶させてから文字式の演算をすると、後で式の確認も容易です。とても便利な使いかたのひとつです。

■例題③



左図のようなRL直列回路に流れる電流は、スイッチSWを閉じてから t 秒後に

$$i(t) = \frac{E}{R} (1 - e^{-\frac{R}{L}t})$$

となります。

今、 $E = 4.5 \text{ [V]}$ $L = 160 \text{ [mH]}$ $R = 55 \text{ [Ω]}$ として

$t = 0.1 \text{ [ms]}$ における電流の値を求めなさい。

このような例題をプログラム化しますと、142ページの例題⑨のようなものが考えられます。

練習問題 2

次の計算式を計算しなさい。ただし、式の中の↓のついているところで、区切って連続計算をする練習を
しなさい。

	計 算 式	キ ー 操 作	答
1	$\frac{1.71 \times 2.43}{3.25 \times 1.35}$ ✓		
2	$\frac{6.48 \times 6.87}{5.13 + 6.03 - 3.41}$ ✓		
3	$\frac{15 \times 3.45}{14.4} + \frac{189 \times 0.58}{8.6}$ ✓		

練習問題 3

次の問題を小数点以下2桁で答えを求めなさい。

	計 算 式	キ ー 操 作	答
	まず、DIGIT (デジット) 指定 を行ってください。		
1	$4 \times (-20) + 5$		
2	$0.95 + \frac{0.79}{3.6}$		
3	$\frac{8.5 \times 10^{20}}{6.24 \times 10^{18}}$		
4	$\frac{5 \times 1.8}{87.93 + 24.15}$		
5	$\frac{1}{1 \times 2} + \frac{2}{2 \times 3} + \frac{3}{3 \times 4}$		

USING (ユージング) 指定でも計算してみましょう。

練習問題 4

	計 算 式	キ ー 操 作	答
1	$(263 + 185)^2$		
2	$15^2 + 38^2 + 73^2 - 56^2 - 24^2$		
3	$13^2 \times 28^2 \div 46^2 \times 89^2 \div 9^2$		
4	$(88^2 + 73^2)^2$		
5	$(2.85 \times 10^3)^2 + (62.98 \times 10^2)^2$		

練習問題 5

	計 算 式	キ ー 操 作	答
1	$\sqrt{34} \times \sqrt{86}$		
2	$\sqrt{53 + 95} + \sqrt{0.84}$		
3	$18 \times \pi \times \sqrt{\frac{843}{257}}$		
4	$\sqrt{12(12-5)(12-4)(12-3)}$		

練習問題 6

	計 算 式	キ ー 操 作	答
1	$(18+7)^3 \times 0.5$		
2	$\left(\frac{5.8 \times 10^2 + 3.8}{7.2 \times \sqrt{105}}\right)^3$		
3	$(3.2 + \sqrt{26.3 \times 8.1})^3$		

練習問題 7

	計 算 式	キ ー 操 作	答
1	$\sqrt[3]{53 \times 0.25 + 72 \times 1.92}$		
2	$\sqrt[3]{3.2 + \sqrt{26.3 \times 8.1}}$		
3	$\sqrt[3]{\frac{980}{4} \left(\frac{6.4 \times 10^8 \times 2.3 \times 10^6}{\pi}\right)^2}$		

練習問題 8

	計 算 式	キ ー 操 作	答
1	$\frac{1}{\sqrt{21 \times 33}}$		
2	$\frac{1}{(87.93 + 24.15) \times (13.84 - 27.65)}$		
3	$\frac{1}{6.24 \times 10^{-10} + 3.85 \times 10^{-10} + \sqrt[3]{1.3 \times 10^{-30}}}$		
4	$\frac{1}{1 \times 2} + \frac{1}{2 \times 3} + \cdots + \frac{1}{n \times (n+1)}$ $= \frac{1}{1 + \frac{1}{n}}$ <p>$n=5, n=10, n=15$のとき 右辺と左辺を別々に計算し等しくなる ことを確かめなさい</p>		

練習問題 9

	計 算 式	キ ー 操 作	答
1	$4^3 + 7^2 + 8^4$		
2	$2.5^3 + 7.3^3$		
3	$4.5^{3.2} + 2.5^{4.3}$		
4	$4.8^{-2} + 8.9^{-3} \times 1980$		
5	$\frac{1}{3.8^{-1.5} + 6.4^{-2.1}}$		

練習問題 14

次の文を読んで計算を行い、階乗の計算に慣れましょう。

問 題 文	計算式とキー操作	答
1 10人の中から4人のリレーランナーを選ぶ場合、 (a) 走る順番を決めてランナーを選ぶには (b) 走る順番を決めないでランナーを選ぶには	${}_{10}P_4 =$ ${}_{10}C_4 =$	
2 12人の人がいます。次のような場合の数を求めなさい。 (a) 1列に並ぶ場合 (b) 円形に並ぶ場合 (c) 特定の4人が隣り合うよう12人が1列に並ぶ場合	$12!$ $(12-1)!$ $9! \times 4!$	

練習問題 15

計 算 式	キ ー 操 作	答
1 $\log 15$		
2 $\log 18.465$		
3 $\log 0.0018465$		
4 $\log 1.84 \times 10^{30}$		
5 $\log \sqrt[3]{\frac{0.95 \times 7.35}{0.625}}$		
6 $3.8 \log 0.49$		
7 $\log \frac{\sqrt{81^2-3}}{15}$		
8 $\log_3 81$ ($\log_a b = \frac{\log b}{\log a}$ を使って)		

練習問題 16

計 算 式	キ ー 操 作	答
1 $10^{3+\log 30}$		
2 $10^{-1.3}$		
3 $10^{14-13.3}$		
4 $10^{-0.012 \times 25 + 2.64}$		

練習問題 17

計 算 式	キ ー 操 作	答
1 $e^{3.8}$		
2 $e^{2 \times (3+7+5)}$		
3 $3.5 \times e^{-\frac{2}{7}}$		
4 $\frac{19.3}{4.8+12.5 \times e^{-3.8}}$		
5 $\frac{15.2}{2.3+8.5 \times e^{-4}}$		

練習問題 18

	計 算 式	キ ー 操 作	答
1	$\cos(0.7\pi)$ (RAD)		
2	$4.5 \sin^2 28^\circ 14' 52''$ (DEG)		
3	$4 \tan\left(\frac{\pi}{3}\right)$ (RAD)		
4	$\frac{\cos 1.286}{\tan 6.254}$ (RAD)		
5	$\sqrt{7} \cos\left(\frac{\pi+3.8}{4.6}\right)$ (RAD)		
6	$4 \cos \frac{40^\circ}{2} \times \cos \frac{60^\circ}{2} \times \cos \frac{80^\circ}{2}$ (DEG)		
7	$\sin 80^\circ + \tan 80^\circ$ (GRAD)		
8	$\sqrt{1 - \cos^2 30^\circ}$ (DEG)		
9	$\cos^2 30^\circ + \sin^2 30^\circ$ (DEG)		
10	$\frac{1}{2} \times 18 \times 15.5 \times \sin 30^\circ$ (DEG)		



角度指定について

角度指定を行ってください。角度指定は **2nd F** **DRG** で行います。

練習問題 19

	計 算 式	キ ー 操 作	答
1	$\sin^{-1} 0.345$ (DEG)		
2	$\cos^{-1} 0.345$ (RAD)		
3	$\tan^{-1}(-4.545)$ (DEG)		
4	$\sin^{-1} 0.56 + 0.57$ (DEG)		
5	$\cos^{-1}(-0.24) - 4.56$ (DEG)		
6	$\sin^{-1} 0.56 + 0.57 \times \cos^{-1}(-0.24) - 4.56 \times \tan^{-1} 0.56$ (DEG)		
7	$\tan^{-1}\left(\frac{3 \sin 60^\circ}{4 + 3 \cos 60^\circ}\right)$ (DEG)		

練習問題 20

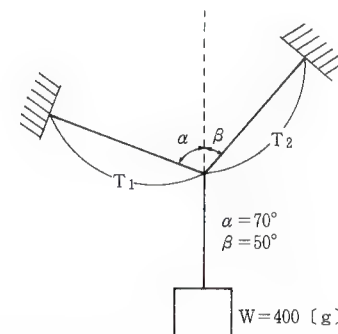
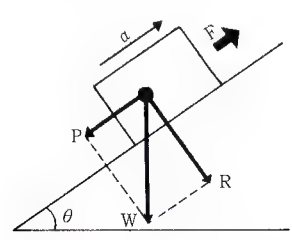
	計 算 式	キ ー 操 作	答
1	直交座標 (10, 10) を極座標 (度) に		
2	直交座標 (32.5, -21.7) を極座標 (度) に		
3	直交座標 (0.35, 0.85) を極座標 (ラジアン) に		
4	極座標 $(15, \frac{\pi}{3})$ を直交座標に (π があるからラジアンです)		
5	極座標 (73, 75°) を直交座標に		
6	極座標 $(8, -\frac{4\pi}{5})$ を直交座標に		

練習問題 21

次の統計計算を行いなさい。

問	題	キ ー 操 作	答																																																
1.	<p>学生の1カ月当りの親からもらうおこづかいを調べたところ、下表のようになりました。平均値と標準偏差（s）を求めなさい。</p> <table><tr><th>お 金</th><th>人 数</th></tr><tr><td>以上 未満 0～ 500</td><td>1</td></tr><tr><td>500～1000</td><td>1</td></tr><tr><td>1000～1500</td><td>2</td></tr><tr><td>1500～2000</td><td>8</td></tr><tr><td>2000～2500</td><td>10</td></tr><tr><td>2500～3000</td><td>10</td></tr><tr><td>3000～3500</td><td>12</td></tr><tr><td>3500～4000</td><td>25</td></tr><tr><td>4000～4500</td><td>11</td></tr><tr><td>4500～5000</td><td>5</td></tr></table>	お 金	人 数	以上 未満 0～ 500	1	500～1000	1	1000～1500	2	1500～2000	8	2000～2500	10	2500～3000	10	3000～3500	12	3500～4000	25	4000～4500	11	4500～5000	5																												
お 金	人 数																																																		
以上 未満 0～ 500	1																																																		
500～1000	1																																																		
1000～1500	2																																																		
1500～2000	8																																																		
2000～2500	10																																																		
2500～3000	10																																																		
3000～3500	12																																																		
3500～4000	25																																																		
4000～4500	11																																																		
4500～5000	5																																																		
2.	<p>下表は、14人の成人男子の身長 x（cm）と体重 y（kg）を測定した結果です。相関係数を求め、体重が58kgの人の身長および身長が178cmの人の体重を推定しなさい。</p> <table><tr><th>No.</th><th>身長</th><th>体重</th><th>No.</th><th>身長</th><th>体重</th></tr><tr><td>1</td><td>171.2</td><td>56.0</td><td>8</td><td>179.0</td><td>67.0</td></tr><tr><td>2</td><td>167.6</td><td>60.5</td><td>9</td><td>163.5</td><td>50.5</td></tr><tr><td>3</td><td>182.5</td><td>92.0</td><td>10</td><td>169.0</td><td>68.0</td></tr><tr><td>4</td><td>175.0</td><td>72.5</td><td>11</td><td>177.0</td><td>65.0</td></tr><tr><td>5</td><td>165.5</td><td>53.0</td><td>12</td><td>174.7</td><td>68.5</td></tr><tr><td>6</td><td>173.8</td><td>64.5</td><td>13</td><td>178.0</td><td>65.0</td></tr><tr><td>7</td><td>170.0</td><td>59.0</td><td>14</td><td>167.8</td><td>62.5</td></tr></table>	No.	身長	体重	No.	身長	体重	1	171.2	56.0	8	179.0	67.0	2	167.6	60.5	9	163.5	50.5	3	182.5	92.0	10	169.0	68.0	4	175.0	72.5	11	177.0	65.0	5	165.5	53.0	12	174.7	68.5	6	173.8	64.5	13	178.0	65.0	7	170.0	59.0	14	167.8	62.5		
No.	身長	体重	No.	身長	体重																																														
1	171.2	56.0	8	179.0	67.0																																														
2	167.6	60.5	9	163.5	50.5																																														
3	182.5	92.0	10	169.0	68.0																																														
4	175.0	72.5	11	177.0	65.0																																														
5	165.5	53.0	12	174.7	68.5																																														
6	173.8	64.5	13	178.0	65.0																																														
7	170.0	59.0	14	167.8	62.5																																														

練習問題 22

問	題	キ ー 操 作 ・ 答
1.	<p>第1図のように、糸に400 gのおもりがつり下げられています。糸の張力 T_1、T_2 を求めなさい。答えは小数点以下2桁とします。</p> <p>ラミーの定理</p> $\frac{W}{\sin(\alpha+\beta)} = \frac{T_1}{\sin(180^\circ-\beta)} = \frac{T_2}{\sin(180^\circ-\alpha)}$ <p>(ヒント)</p> <p>小数点以下2桁指定はDIGIT指定で行います。</p> <p>第1図</p> 	
2.	<p>第2図のように、重さ20 [kg] の物体を角度35°の斜面上にそって5 [m] を加速度8 [m/sec²] で引き上げるのに4秒かかりました。この場合の仕事と仕事率を求めなさい。</p> <p>$R = W \cos \theta$ $P = W \sin \theta$ $W = 20$ [kg]、$\theta = 35^\circ$、$S = 5$ [m] $a = 8$ [m/sec²]、$t = 4$ [sec] $g = 9.8$ [m/sec²]、$\mu = 0.3$ 力: $F = W \left(\frac{a}{g} + \mu \cos \theta + \sin \theta \right)$</p> <p>仕事: $W_p = F \cdot S$ 仕事率: $P_t = F \cdot S / t$</p> <p>第2図</p> 	

3. 直径3 [cm]、長さ50 [cm] の銅棒の両端を固定し、棒の温度を20°Cから70°Cまで加熱して上昇させたとき、棒に生じる応力および、固定端の圧力を求めなさい。

ただし、

銅の線膨張係数 α は $\alpha = 0.167 \times 10^{-4}$ [°C⁻¹]

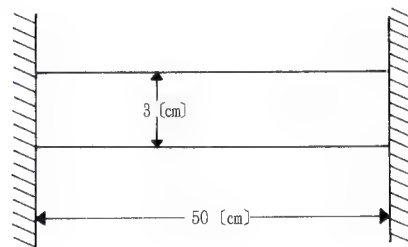
銅の縦弾係数 E は $E = 1.1 \times 10^8$ [kg/cm²]

とします。

熱応力: $\delta c = E \cdot \alpha (t' - t)$

圧力: $F_c = A \cdot E \cdot \alpha (t' - t)$ ($\because A = \pi r^2$)

第3図



4. 頂角28°の直角形くさびを木材に打ち込むのに100kg重の力を要しました。木材を引きさく力を求めなさい。ただし $\mu = 0.25$ とします。

$$2\alpha = 28^\circ$$

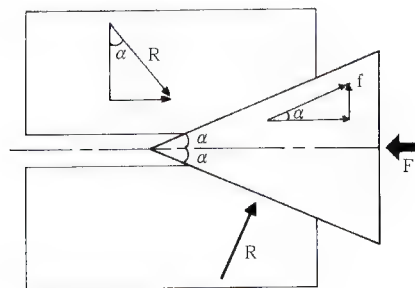
$$F = 100 \text{ [kg重]}$$

$$\mu = 0.25$$

$$\text{摩擦角: } \phi = \tan^{-1} \mu$$

$$\text{引きさく力: } R = \frac{F \cdot \cos \phi}{2 \sin(\alpha + \phi)}$$

第4図



5. A点に下向きの力Pがかかるとき、 N_1 、 N_2 がどのような力であればつりあいますか。

$$\Sigma X = 0 \text{ (X方向の分力の和が0)}$$

$$\Sigma Y = 0 \text{ (Y方向の分力の和が0)}$$

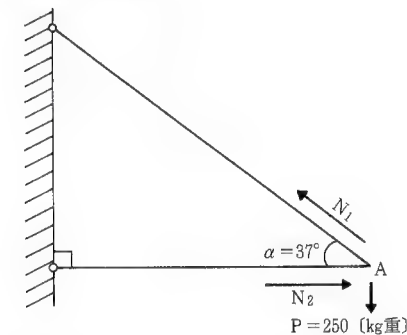
$$\Sigma X = 0 \text{ より } -N_1 \cos \alpha + N_2 = 0$$

$$\Sigma Y = 0 \text{ より } N_1 \sin \alpha - P = 0$$

$$\text{すなわち } N_1 = \frac{P}{\sin \alpha}$$

$$N_2 = N_1 \cos \alpha$$

第5図



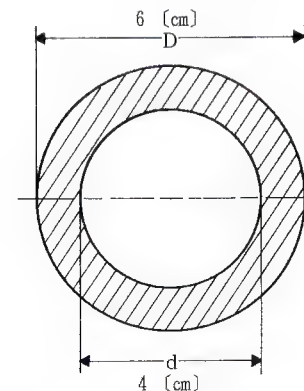
6. 第6図のような外径D、内径dをもつ断面について、断面積A、重心軸に対する2次モーメントI、および断面係数Zを求めなさい。

$$\text{断面積} : A = \frac{\pi}{4} (D^2 - d^2) \text{ [cm}^2\text{]}$$

$$\text{断面2次モーメント: } I = \frac{\pi}{64} (D^4 - d^4) \text{ [cm}^4\text{]}$$

$$\text{断面係数} : Z = \frac{\pi}{32} \cdot \frac{D^3 - d^3}{D} \text{ [cm}^3\text{]}$$

第6図



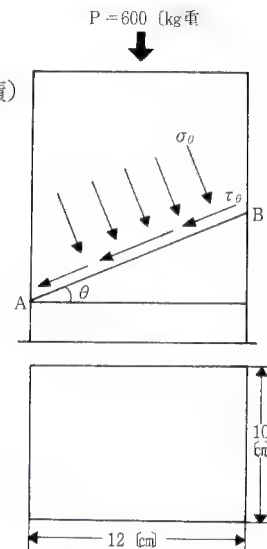
7. 第7図のような場合で、
 θ が 15° 、 30° 、 45° 、 60°
 のとき、A B面に生じる
 垂直応力度 σ_θ 、せん断
 応力度 τ_θ を求めなさい。

第7図

$$\sigma = \frac{P}{A} \text{ [kg重/cm}^2\text{]} \quad (A = \text{面積})$$

$$\sigma_\theta = \sigma \cos^2 \theta \text{ [kg重/cm}^2\text{]}$$

$$\tau_\theta = \frac{\sigma}{2} \sin 2\theta \text{ [kg重/cm}^2\text{]}$$

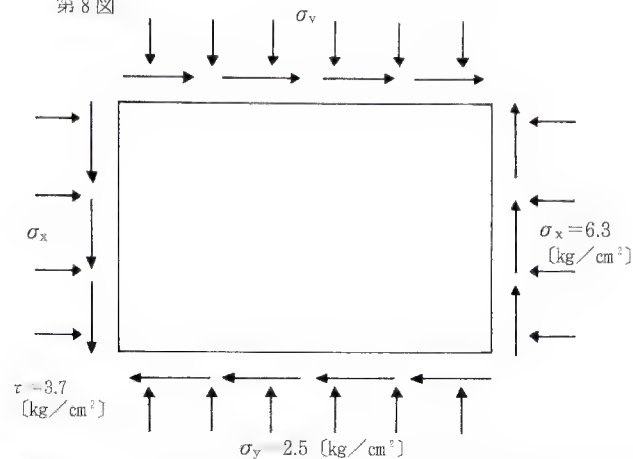


8. 構造物内部の微小部分が第8図のような応力度を受ける場合、主応
 力度およびその方向を求めなさい。

$$\text{主応力度の角 } \tan 2\theta = \frac{2\tau}{\sigma_x - \sigma_y}$$

$$\text{主応力度 } \sigma_{I, II} = \frac{1}{2}(\sigma_x + \sigma_y) \pm \sqrt{\frac{1}{4}(\sigma_x - \sigma_y)^2 + \tau^2}$$

第8図



9. 第9図のような3つの力 P_1 、 P_2 、 P_3 の合力を求めなさい。

合力の大きさ

$$R = \sqrt{\Sigma X^2 + \Sigma Y^2}$$

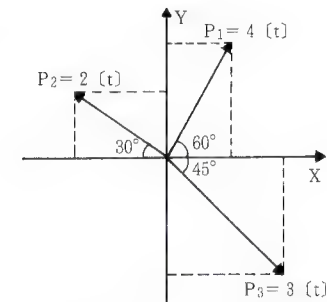
合力の向き

$$\theta = \tan^{-1} \frac{\Sigma Y}{\Sigma X}$$

$$\Sigma X = P_1 \cos \theta_1 + P_2 \cos \theta_2 + P_3 \cos \theta_3 \quad \left(\begin{array}{l} \theta_1 = 60^\circ \\ \theta_2 = 180^\circ - 30^\circ \\ \theta_3 = 360^\circ - 45^\circ \end{array} \right)$$

$$\Sigma Y = P_1 \sin \theta_1 + P_2 \sin \theta_2 + P_3 \sin \theta_3$$

第9図



10. 幅 8 [m]、水深 3 [m] の長方形水路の動水こう配が
 $\frac{1}{1500}$ のとき、平均流速、流量を求めなさい。ただしマンニングの粗度
 係数を $n = 0.02$ とします。

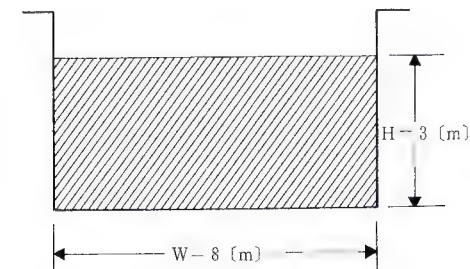
$$\text{流 積 } A = W \cdot H \text{ [m}^2\text{]}$$

$$\text{潤 辺 } S = W + 2 H \text{ [m]}$$

$$\text{平均流速 } V = \frac{1}{n} \left(\frac{A}{S} \right)^{\frac{2}{3}} \sqrt{I} \text{ [m/sec]} \quad \left(\begin{array}{l} I = \frac{1}{1500} \\ n = 0.02 \end{array} \right)$$

$$\text{流 量 } Q = V \cdot A \text{ [m}^3\text{/sec]}$$

第10図



11. 第11図のように掘削した斜面の下に、軟弱な土質があることがわかっています。軟弱な土質の内部
 摩擦角 $\phi' = 5^\circ$
 粘着力 $C' = 2 \text{ [t/m}^2\text{]}$
 としてこの斜面のブロックすべりに対する安全率を求めなさい。

斜面の土の内部摩擦角 $\phi = 20^\circ$
 斜面の土の粘着力 $C = 0$
 斜面の土の単位体積重量 $\gamma = 1.8 \text{ [t/m}^3\text{]}$ とします。

- 左向きにすべらせようとして働く主動土圧: P_A

$$P_A = \frac{\gamma H_1^2}{2} \tan^2(45^\circ - \frac{\phi}{2}) \quad [\text{t/m}]$$

- すべりを止めようとして右向きに働く受働土圧: P_P

$$P_P = \frac{\gamma H_2^2}{2} \tan^2(45^\circ + \frac{\phi}{2}) \quad [\text{t/m}]$$

- 粘着力による抵抗: $C' L \quad [\text{t/m}]$

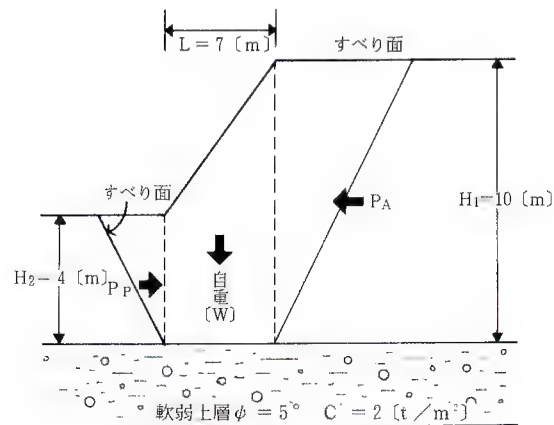
- 摩擦力による抵抗: $W \tan \phi = \frac{H_1 + H_2}{2} L \cdot \tan \phi' \quad [\text{t/m}]$

- 斜面のブロックすべりに対する安全率: F_s

$$F_s = \frac{C' L + W \tan \phi + P_P}{P_A}$$

$$F_s = \frac{C' L + \frac{H_1 + H_2}{2} L \tan \phi' + \frac{\gamma H_2^2}{2} \tan^2(45^\circ + \frac{\phi}{2})}{\frac{\gamma H_1^2}{2} \tan^2(45^\circ - \frac{\phi}{2})}$$

第11図



12. 背部の地表面が水平な内部摩擦角 $\phi = 30^\circ$ の土砂からなるがけに、高さ6メートルの第12図のような壁面（土と壁の摩擦角 $\delta = 20^\circ$ ）を有する重力式擁壁を作りたい。
 土の単位体積重量 $\gamma = 1.8 \text{ [t/m}^3\text{]}$ として主動土圧 P_A を求めなさい。

$$\theta = 90^\circ$$

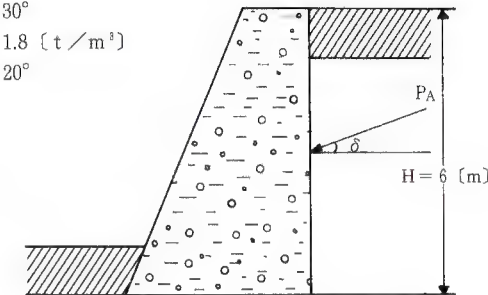
$$\beta = 0^\circ$$

$$\phi = 30^\circ$$

$$\gamma = 1.8 \text{ [t/m}^3\text{]}$$

$$\delta = 20^\circ$$

第12図



クーロン土圧の解析公式

$$P_A = \frac{\gamma H^2}{2} \cdot \frac{\sin^2(\theta + \phi)}{\sin^2 \theta \sin(\theta - \delta) \left\{ 1 + \sqrt{\frac{\sin(\phi + \delta) \sin(\phi - \beta)}{\sin(\theta - \delta) \sin(\theta + \beta)}} \right\}^2} \quad [\text{t/m}^2]$$

13. 第13図の回路において端子A B間、および端子C D間で測定した合成抵抗を求めなさい。

$$R_1 = 400 \text{ [}\Omega\text{]}$$

$$R_2 = 730 \text{ [}\Omega\text{]}$$

$$R_3 = 240 \text{ [}\Omega\text{]}$$

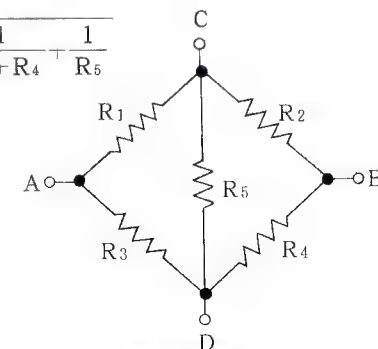
$$R_4 = 438 \text{ [}\Omega\text{]}$$

$$R_5 = 600 \text{ [}\Omega\text{]}$$

$$R_{AB} = \frac{(R_1 + R_2)(R_3 + R_4)}{R_1 + R_2 + R_3 + R_4} \quad (\text{ブリッジが平衡しているとき})$$

$$R_{CD} = \frac{1}{\frac{1}{R_1 + R_3} + \frac{1}{R_2 + R_4} + \frac{1}{R_5}}$$

第13図



14. ある工場の三相負荷 P は 20 [kW]、力率 60% である。これを力率 80% に改善するために要するコンデンサ Q [kVA] を求めなさい。

また、コンデンサにかかる電圧を 200 [V] としたときの静電容量 C [μ F] を求めなさい。ただし、周波数は 50 [Hz] としなす。

改善前の力率角 θ_1 、改善後の力率角 θ_2 とするとコンデンサ Q は

$$Q = P (\tan \theta_1 - \tan \theta_2)$$

静電容量は

$$3 C = \frac{Q}{\omega \cdot V^2}$$

$$(\because \omega = 2\pi f)$$

15. 第15図のような RLC 直列回路において、 $R = 300$ [Ω] について、スイッチ SW を閉じてから $t = 5$ [ms] 後の電流値を求めなさい。

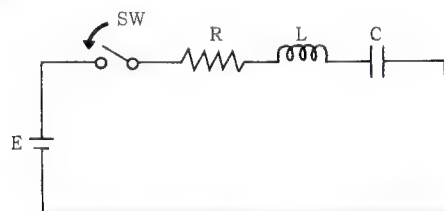
● $R < 2 \times \sqrt{\frac{L}{C}}$ の場合 (振動的減衰)

$$\alpha = \frac{R}{2L}$$

$$\beta = \frac{1}{2L} \sqrt{\frac{4L}{C} - R^2}$$

$$i = \frac{E}{\beta L} e^{-\alpha t} \cdot \sin \beta t$$

第15図



$$L = 150 \text{ [mH]}$$

$$C = 2 \text{ [\mu F]}$$

$$E = 100 \text{ [V]}$$

16. (a) 試薬特級の無水炭酸ナトリウム 5.2050 g を秤量し、溶解したものをメスフラスコで 1 l に希釈し標準溶液とした。この規定度を求めなさい。

- (b) ある井戸水を分析したところ、 100 ml 中に CaSO_4 2.8 mg、 $\text{Mg}(\text{HCO}_3)_2$ 7.5 mg が含まれていた。この水の硬度を求めなさい。

ただし、 $H = 1.008$ $C = 12.01$ $O = 16$

$Mg = 24.3$ $S = 32.06$ $Ca = 40.08$

$Na = 22.99$

とする。

- (a) Na_2CO_3 の 1 g 当量 $= \text{Na}_2\text{CO}_3 / 2$ である。

- (b) 水 100 ml 中の Ca 塩および Mg 塩のモル数の合計を CaO の質量 n mg に換算し、 n° とする。

17. ある実験で二酸化炭素 1 mol が、 40°C 、 50 atm で 0.380 l の容積を占めることが実測された。その圧力を、

(a) 理想気体の状態式で、

(b) ファンデルワールスの状態式で

計算し、実測値と比較しなさい。

$$(a) PV = RT \quad \rightarrow \quad P = \frac{RT}{V}$$

$$(b) \left(P + \frac{a}{V^2}\right)(V - b) = RT \rightarrow P = \frac{RT}{V - b} - \frac{a}{V^2}$$

$$R = 0.0821 \text{ (l} \cdot \text{atm} / (\text{mol} \cdot \text{K}))$$

$$a = 3.60 \quad b = 4.28 \times 10^{-2}$$

18. 次の水溶液について質問の値を求めなさい。

- (a) 0.01N-HCl 水溶液の pH
 (b) 0.05N-NaOH 水溶液の pH
 (c) pH=2.50の H_2SO_4 20ml を水でうすめて 500ml とした水溶液の pH
 (d) pH=8.50 の水溶液の $[\text{H}^+]$

HCl、NaOH は完全電離とする。

$$\text{pH} = -\log \frac{K_w}{[\text{OH}^-]}$$

$[\text{H}^+]$: 水素イオン濃度 (mol/ℓ)
 $[\text{OH}^-]$: 水酸イオン濃度 (mol/ℓ)

$[K_w]$: 水のイオン積 $1.0 \times 10^{-14} (\text{mol}/\ell)^2$

19. 次の実験データがある。この分解反応が 1 次であることを示し、かつ、この温度における速度定数を求めなさい。ただし、初濃度 $a = 1.590 (\text{mol}/\ell)$ とします。

実験番号	1	2	3	4
時間 t (min)	5.4	21.6	25.5	32.9
変化量 x (mol/ℓ)	0.624	1.298	1.376	1.474

反応温度 40°C

1 次反応ならば $\frac{dx}{dt} = k(a-x)$

k : 速度定数 (min^{-1})

$t = 0$ のとき $x = 0$ であるから

$$k \int_0^t dt = \int_0^x \frac{dx}{a-x} \quad \text{より} \quad k = \frac{1}{t} \ln \frac{a}{a-x}$$

各実験番号における k を計算し、この値がほぼ一定であることを確かめてから平均値を求める。

20. 円管のまざつ係数 (f) とレイノルズ数 (Re) との関係式は次のようになります。

$$1/\sqrt{f} = A \log(Re \sqrt{f}) + B$$

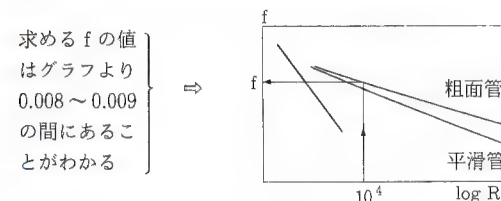
ただし、

	A	B
平滑管	4.0	-0.4
粗面管	3.2	1.2

(化工便覧)

いま、粗面管において $Re = 10000$ のときの f を試算法で求めなさい。

● (左辺)-(右辺) がゼロになるようにグラフで解く。



参 考

計算の優先順位

本機はカッコ、関数を含めて数式どおりのキー操作で計算を行うことができます。

計算の優先順位の判断や途中結果の処理はすべて計算機が自動的に処理してくれます。計算の優先順位は次のとおりです。

1. π や変数の呼び出し
2. 関数 (SIN、COS など)
3. べき乗 (\wedge)
4. 符号 (+、-)
5. 乗除算 (*、/)
6. 整数の除算 (\div)
7. 整数の剰余 (MOD)
8. 加減算 (+、-)
9. 大小比較 (>、>=、<、<=、<>、=)
10. AND、OR、XOR

(注) ● カッコが使用されている場合はカッコ内の計算が最優先されます。

● 複合関数 ($\sin \cos^{-1} 0.6$ など) は右から左の順で計算されます。

● べき乗の連算 (3^{4^2} すなわち $3 \wedge 4 \wedge 2$ など) は右から左の順で計算されます。

● 上記 3. と 4. では後に出てきたほうが優先順位が高くなります。

〈例〉 $-2 \wedge 4$ は $-(2^4)$ となります。

$3 \wedge -2$ は 3^{-2} となります。

第4章

BASIC言語

1. BASIC言語をマスターする第一歩

(1) BASICとは

B ; Beginner's (ビギナーズ) 初心者向きで
 A ; All-Purpose (オールパーパス) あらゆる目的に合う
 S ; Symbolic (シンボリック) 記号を使った
 I ; Instruction (インストラクション) ... 命令
 C ; Code (コード) 語

それぞれの頭文字を取って、BASIC (ベーシック) という名のついたコンピュータ言語です。BASIC はやさしい英語と記号でできており、コンピュータがどんな仕事をすればよいのかを示す一連の命令文、すなわち「プログラム」を人間とコンピュータが対話するような形で作っていけるのが大きな特長です。

(2) プログラムとは

プログラムとは、計算機 (コンピュータ) に計算などを行うための手順を指令する命令書のようなものです。この指令をコンピュータが理解できるように書き表したり、コンピュータに記憶させたりすることをプログラミングとか、プログラムを組むといいます。コンピュータでいろいろな問題を処理しようとするとき、最初からBASIC言語でプログラミングする方法もありますが、ここではBASIC言語のプログラムをマスターする第一歩として、問題を処理する方法の考えかたや、手順を整理し、図的に順序だてて、プログラムの流れをわかりやすく表現することから始めます。

(3) 主な記号の読みかたと意味

記号	読みかた	意 味	記号	読みかた	意 味	記号	読みかた	記号	読みかた
+	プラス	加 算	「	ダブルクォーテーション	引用符	!	イスクラメーションマーク	[]	大カッコ
-	マイナス	減 算	,	ピリオド	小数点	?	クエッションマーク	{ }	中カッコ
*	アスタリスク	乗 算	,	コンマ	区切り	@	アットマーク	¥	円記号
/	スラッシュ	除 算	:	コロン	命令文を続けて書く	&	アンドマークまたはアンパサンド	'	パイプ記号
=	イコール	等 号	;	セミコロン	表示部をつめる	%	パーセントマーク	~	波形
>	グレーター・ザン	より大きい	^	ベキ (ハットマーク)	べき乗記号	\$	ドルマーク	—	アンダーライン
<	レス・ザン	より小さい	#	クロスハッチ	番号記号	'	シングルクォーテーション		

(4) 「流れ図」 (フローチャート Flowchart) について

プログラムの処理方法の考えかたや手順を整理し、図的に順序だてて書いたものを「流れ図」または「フローチャート」といいます。

流れ図は、図記号と簡単な式や文字で表します。図記号はそれぞれ意味を持っています。次に代表的な図記号とその意味、そして流れ図の基本形を示しますので十分に理解してください。

(5) 代表的な「流れ図」とその意味

流 れ 図 の 記 号	意 味
1.	端 子 (Terminal Interrupt) 流れ図の開始、終了などの端子を表します。
2.	処 理 (Process) 枠内 (枠外に書いてもよい) に書かれてある処理を行います。
3.	準 備 (Preparation) 初期値などの準備などに用います。
4.	入 出 力 (Input/Output) 情報の入出力を意味します。入出力一般として用いられます。また、本書では図記号 をRESTOREとみなします。
5.	手操作入力 (Manual Input) 変数への入力など、キーボードなどから手で操作して入力することを表します。
6.	定義済み処理 (Predefined Process) サブプログラムなど、別の場所で定義されている命令群などの処理を表します。
7.	表 示 (Display) 情報を人間が利用できるように、ディスプレイに表示します。
8.	判 断 (Decision) 判断・比較を行います。

9.		書類 (Document) 書類を媒体とする入出力機能を表します。
10.		繰り返しループ ループの始まりと終わりを表します。
11.		結合子 (Connector) 流れ図のほかの場所への出口、または、ほかの場所からの入口を表します。
12.		流れ線 (Flow Line) 記号を結びつける機能を表します。
13.		注釈 (Comment Annotation) 明りょうにするために、説明または注意を加える機能を表します。

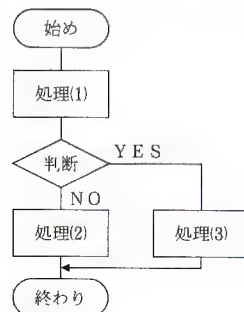
(6)プログラムの形 (構造)

1. 直線形の流れ図 (シーケンス構造)



- 直線形プログラム
- 「判断」が入っていません。

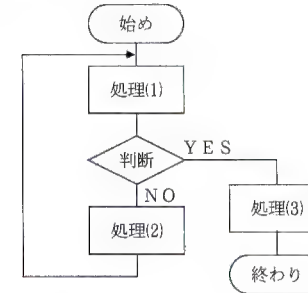
2. 分岐形の流れ図 (IF~THEN構造)



- 分岐形プログラム
- 設定された条件によってプログラムの流れが分岐します。
- もし、条件が満たされたら 処理(3)を行い、そうでなければ、処理(2)を行います。

3. ループ形の流れ図 (ループ構造)

〈その1〉



- 繰り返し形プログラム 〈その1〉
- 条件が満たされるまで繰り返し、処理(1)と処理(2)を行います。条件が満たされると、ループを抜けて処理(3)を行います。

〈その2〉



- 繰り返し形プログラム 〈その2〉
- 条件が満たされている間は、処理(1)を繰り返します。
- 条件が満たされなくなると、ループを抜けて処理(2)を行います。

(7)「流れ図」全体の約束について

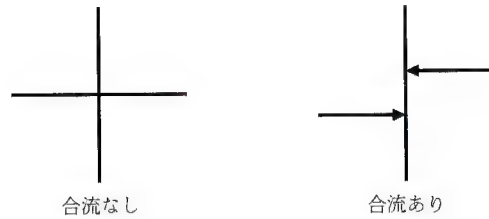
①流れ図における流れ線の方向は、原則として、



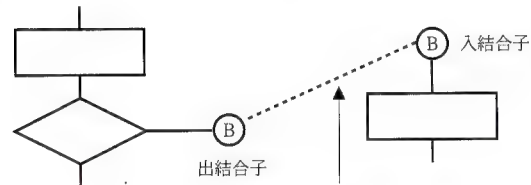
上から下へ 左から右へです。

流れの方向がこれに合わないときは、流れを示す矢印を用います。

②流れ線は、交差してもかまいません。交差しても論理的関係はありません。



③結合子は、流れ線が中断される点を表すのに用いる出結合子と、中断された流れ線が再開される点を表すために用いる入結合子があります。それぞれの結合子に同じ文字や数字などを記入し、これらが結合していることを示します。



(実際は点線など書きませんが) ⑤と⑥で結合していることを示します。

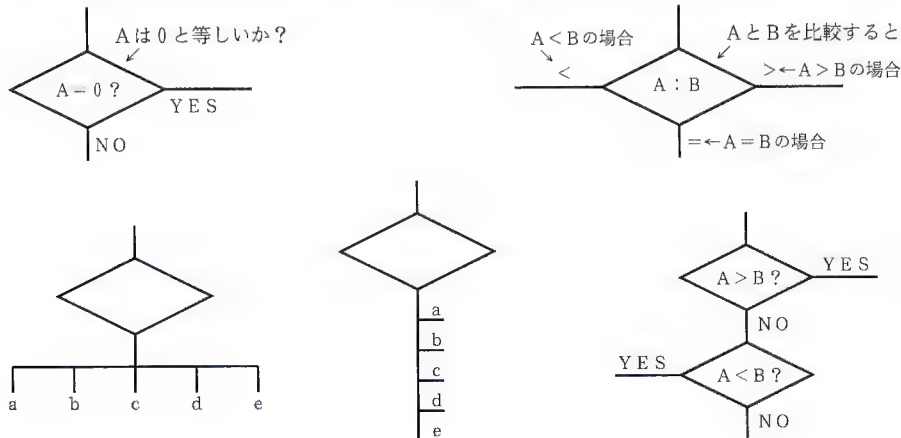
④1つの流れ図記号から出口を2つ以上書く場合は、必要な数だけの流れ線をその記号から出すか、あるいは、その記号から出た流れ線の数だけ分岐する形で表現します。

分岐の出口には、分岐条件を記入します。

「>、=、≠、<」などや「YES」、「NO」あるいは「0、1、+、-」などがよく用いられます。

比較記号としては「:」や「?」などが使われます。

流れ図記号の中だけでなく、図記号の周辺に文章で説明する方法なども多く見受けられます。



プログラムがだんだん複雑になってくると、流れ図を書かないとプログラムの組み立てが困難になります。プログラムを組んでいるときは理解していても、しばらく時間が経過した後、あらためて解読してみるとなかなか手こずることがあります。流れ図の考えかたを早くマスターし、プログラムの基本をしっかりと身につけましょう。

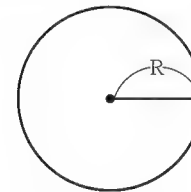
2. プログラムの基本

STEP 1 INPUT、PRINT、END、GOTO文

【例題】①

半径を入力して、円の面積を求める
プログラムを作りなさい。

■解 説



左図において、半径をR、面積をSとすると

$$S = \pi R^2$$






























で面積を求めることができます。














流 れ 図 (フローチャート)	行 番 号 (ラインナンバー)	BASIC言語によるステートメント (ひとつの意味をもった処理式や、命令語のこと)
始め		[BASIC] を押して画面右側に“PRO”を点灯させます。
Rを入力する	10	INPUT R
$S \leftarrow \pi R^2$	20	$S = P I * R \wedge 2$ $S = P I * R * R$ でもよい
面積Sを表示	30	PRINT S
終わり	40	END

(注) π は P I に変換されて入力されます。

(1)プログラムの入力

プログラムを計算機に書き込むときは、次の手順で行います。

操 作 の 手 順	操 作 手 順 の 説 明
① ON を押して電源を入れます。 BASIC でプログラムモード (PROモード) にします。	プログラムを書き込める状態にします。
② NEW	先に入力した (かもしれない) プログラムやデータをすべて消します。
③ 行番号を入れ、文を入れます。 10 INPUT R 行番号とは → (行番号はラインナンバーともいいます) 表示について →	   +    プログラムの実行順序の番号です。 数字の小さい行番号から実行されるので、実行の順に数字の小さい行番号をつけます。 行番号は、10番おきにつけるのが一般的です。後から、プログラムを追加するときに都合がよいからです。 なお、本機は1～65279までの整数を行番号として使用できます。また、行番号はAUTO (オート) 命令 (317ページ参照) を使うと自動発生させることができます。 行番号に続けて文 (ステートメント) を入れ、  を押すと、行番号の後に自動的に: (コロン) が入ります。 10: INPUT R ↑ ↑ 自動的に1桁 (1文字) あきます。 自動的に: (コロン) が入ります。
④ 20 S=PI*R^2 (S=PI*R*Rでも同じです。)	行番号10で与えられたRの値で $\pi \cdot R^2$ を計算し、計算した数値をSという記号の中に入れておきなさい (という意味です)。    =  *    
⑤ 30 PRINT S	   +    行番号20のSの記号の中に入っている数値を表示しなさい (という意味です)。
⑥ 40 END ENDの意味 →	      これでプログラムは終了しました (という意味です)。
 について	 は文の終わりに必ず入力します。 これは「この文は終わりました」または「プログラム文として書き込むことが終了しました」という意味が含まれています。

INPUTの入力方法 →	 +  (    でも同じです。)
PRINTの入力方法 →	 +  (    でも同じです)。 このように、本機ではBASIC言語の命令語が、いくつか準備されていますので、活用してください。
文 (ステートメント) について	1つの行 (ライン) は1つ以上の文 (ステートメント) からなり、2つ以上の文 (マルチ・ステートメント) になる場合は文と文の間に: (コロン) を入れて区別します。 (コロンを入れる) 行番号 文 : 文 文  [例題] ①をマルチ・ステートメントにする 10 INPUT R: S=PI*R^2: PRINT S: ENDとなります。ただし1行254文字以上は書けません。









NEWについて

NEW (ニュー) とは「新しい」や「初めての」という意味です。この命令は、計算機に記憶されている内容を全て消去し、初めの状態にするときに使用します。プログラムを新しく入れる前に必ず操作するBASIC命令です。


(2)プログラムの修正・編集

プログラムを作成する場合、キー操作のミスなどにより正しいプログラムにならないことが多くあります。

このような場合は下表のキー操作で修正・編集します。

表示されている行 (ライン) 内の	 または  でカーソルを移動し、正しいキーの入力
① 訂 正	
② 削 除 (Delete デリート) (Backspace バックスペース)	 +  
③ 追 加 (Insert インサート)	

(3)プログラムの確認

プログラム全部の内容を計算機に書き込んだら、次にPROモードで下表のキー操作により、その内容の確認をします。 を押すとプロンプト記号 (>) だけの表示になります。

キ ー 操 作	説 明
	<ul style="list-style-type: none"> ・プロンプト記号(>)が表示されているときに押すと、先頭のラインから、画面に表示できる範囲を表示します。 ・1回押すと、現在表示されている次のラインを表示します。押し続けると、順次、次のラインを表示します。 ・画面にカーソルが出ているときは、カーソルを次の行へ移します。
	<ul style="list-style-type: none"> ・プロンプト記号(>)が表示されているときに押すと、最終のラインを表示します。 ・1回押すと、現在表示されている前のラインを表示します。押し続けると、順次、前のラインを表示します。 ・画面にカーソルが出ているときは、カーソルを前の行へ移します。
LIST または L.	先頭のラインから、画面に表示できる範囲を表示します。
LIST 30 または L. 30	行番号(ラインナンバー) 30行から、画面に表示できる範囲を表示します。(指定した行番号および、それより大きい行番号が存在しないときはエラー40になります)
プログラムを呼び出した後、 または を押せば表示部の1行目に表示されているプログラムラインにカーソルが表示されます。 を押し続ければそのプログラムラインの最後までカーソルが移動します。 の場合は先頭まで戻ります。	

(4)行(ライン)の追加・変更・削除

下の表の右側のような誤りがあったとします。

正しいプログラム	誤って入力したプログラム
<pre> 10 INPUT R 20 S= PI *R^2 30 PRINT S 40 END </pre>	<pre> 10 INPUT R 20 PRINT S 30 END </pre> <p><ここに S=PI*R^2がない</p>

誤って入力したプログラムの行番号10行と20行の間に $S = \text{PI} * R^2$ を入れるために、新しく15という行番号を選びます。
15 S=PI*R^2 とすると、下表のように訂正されたプログラムができあがります。

プログラムの内容は同じです。行番号が異なっても問題はありません。

```

10 INPUT R
15 S= PI *R^2
20 PRINT S
30 END

```

これで正しいプログラムが完成しました。

行番号のつけ直し

訂正されたプログラムでは、行番号が10番ごとにはなっていません。

行番号のつけ直しはRENUM(リナンバー)命令(356ページ参照)を使うと簡単に行えます。

- ① プログラムモードで、RENUM と操作します。
- ② を押すとプログラムリストが表示されます。

```

10 INPUT R
15 S= PI *R^2
20 PRINT S
30 END

```

```

10 INPUT R
20 S= PI *R^2
30 PRINT S
40 END

```

追加・変更・削除を行うときは、145ページの「スクリーンエディタについて」も参照してください。

(5)プログラムの実行

プログラムの実行は、実行モード(RUNモード)で行いますので **BASIC** を押して“RUN”を表示させてください。

```

RUN MODE          RUN
>

CAPS
DEG

```

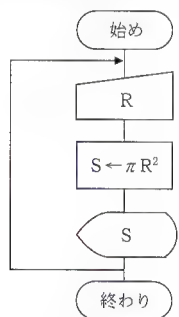
そして次の命令で実行を開始します。

- RUN 最も小さい数字の行番号より実行を開始します。
 RUN 行番号 指定した行番号より実行を開始します。

それでは円の面積を求めるプログラムを実行させてみましょう。

実行の内容	キー操作	表示部
RUNモードに設定		>
実行	(SHIFT) + (V) (RUN) または (R) (U) (N) (ENTER)	?
半径R = 1 [cm] の場合	1 (ENTER)	1 _ 3. 1 4 1 5 9 2 6 5 4 >
半径R = 2 [cm] の場合	(SHIFT) + (V) (RUN) 2 (ENTER)	? 2 _ 1 2. 5 6 6 3 7 0 6 1 >
半径R = 3 [cm]、半径R = 4 [cm] の場合も計算してください。		

このようにしていくと、実行のたびに (SHIFT) + (V) (RUN) と入力するのが、わずらわしくなってきます。そこで下の流れ図のように、Sを表示した後に再び先頭に戻る命令語があれば、入力が簡単になります。BASIC言語では、このようなときにGOTO文を用います。



```

10 INPUT R
20 S = PI * R ^ 2
30 PRINT S
35 GOTO 10
40 END
  
```

30行と40行の間に新しくGOTO文を入れるために
35 GOTO 10 (ENTER) と操作します。

このプログラムを実行すると、簡単に計算を進めることができます。

実行の内容	キー操作	表示部
RUNモード	RUN (ENTER)	?
R = 1 の場合	1 (ENTER)	1 _ 3. 1 4 1 5 9 2 6 5 4
次のRの入力待ち →		?
R = 2 の場合	2 (ENTER)	2 _ 1 2. 5 6 6 3 7 0 6 1
次のRの入力待ち →		?

このプログラムの実行を中止するときは (BREAK) (ON) を押してください。

(6) プログラムのアプリケーション

今までのプログラムの実行は入力・出力にただ数値が入っているだけで、ちょっと目を離したりすれば、それがどのような意味をもつ数値かわからなくなります。

ここではINPUT文とPRINT文に変化をもたせ、わかりやすい工夫を試みることにします。

```

10 INPUT R
10 INPUT 'ハンケイ=R=' ; R
  
```

※この部分をメッセージとして表示します。

セミコロン

メッセージを表示するため、何の数値を入力すればよいのかわかりやすくなります。

スペース

文字と記号を分離してわかりやすいようにするために1文字空白を作りました。
スペースは (SPACE) で入れます。

※ 'ハンケイ=R=' のように、引用符 (ダブルクォーテーション) で囲まれたものは、数値と異なり文字として扱われます。

```

30 PRINT S
30 PRINT 'R=' ; R ; 'S=' ; S
  
```

スペースを入れないと、Rの数値とSの文字がくっつきすぎてわかりにくくなります。スペースを2つ入れて、Rの数値とSの文字を離しておきます。

***** メッセージの表示について *****

' ' で囲まれた中に入れたアルファベットの小文字は、そのまま小文字として扱われます。

' ' で囲まれた中以外でアルファベットの小文字を用いると、大文字に変換されます。ただし、注釈文 (355ページREM命令参照) は除く。

プログラムを整理すると下のようになります。

```

10 INPUT 'ハンケイ=R=' ; R
20 S = PI * R ^ 2
30 PRINT 'R=' ; R ; 'S=' ; S
35 GOTO 10
40 END
  
```

これを実行してみましょう。

実行の内容	キー操作	表示部
RUNモード	RUN	ハンケイ R=_
R=1 次のRの入力待ち →	1 	ハンケイ R=1 _ R=1. S=3. 141592654 ハンケイ R=_
R=2 次のRの入力待ち →	2 	ハンケイ R=2 _ R=2. S=12.56637061 ハンケイ R=_

(7)エラー表示とその処理方法

プログラムを実行したときにプログラムに誤りがあったり、データが不適当な場合などでエラーが発生することがあります。

エラーが発生すると次のようなエラーメッセージが表示されます。

```
ERROR 10 IN 20
```

これは、行番号20行にエラーコード10の内容のエラーが発生したことを意味します。(くわしくは386ページのエラーコード表を参照)

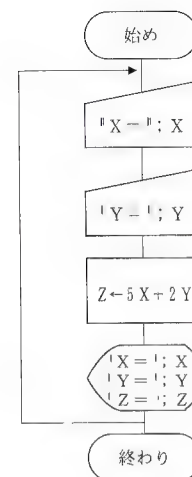
●エラーの処理手順

- ① **[CLS]** でエラーを解除します。
- ② **[BASIC]** でPROモードにします。
- ③ **[▼]** または **[▲]** を押します。
- ④ エラーの発生した行番号が表示され、エラーの発生した位置にカーソルが表示されます。
- ⑤ エラーの発生した原因を探して、プログラムを修正します。

【例題】②

ある適当な2つの数、XとYを入力して、 $5X + 2Y$ を求めるプログラムを作りなさい。

■フローチャート



プログラムを入力する前に **[BASIC]** でPROモードにし、**[NEW]** と操作します。

■プログラム例

```

10 INPUT "X="; X
20 INPUT "Y="; Y
30 Z = 5 * X + 2 * Y
40 PRINT "X="; X; "Y="; Y; "Z="; Z
50 GOTO 10
60 END
  
```

メモリ内容	
変数	内容
X	Xの値
Y	Yの値
Z	$Z = 5X + 2Y$

■数値代入例

Xの値	Yの値	Zの値(答)
1	2	9
5	-3	19
10	50	150
22	36	182

演習問題 1

を行ってください。

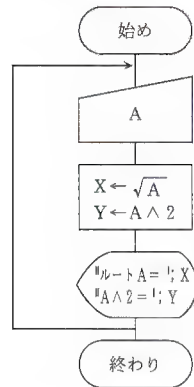
STEP 2 切り捨て・四捨五入・桁指定

プログラムを実行し数値を求める場合、小数点以下2桁程度まで求めればよい場合が多くあります。このようにとき、切り捨て・四捨五入あるいは桁指定をして必要な桁数だけ表示するプログラムについて次の例題で練習してみましょう。

【例題】③

数Aを入力して \sqrt{A} 、 A^2 の値を表示するプログラムを作りなさい。
数Aは1、2、3、4、5.23を入力するものとします。

■フローチャート



■プログラム例

```

10 INPUT A
20 X=SQR A
30 Y=A^2
40 PRINT 'ルートA='; X; 'A^2='; Y
45 GOTO 10
50 END

```

■キー操作手順

操 作	表 示 例
BASIC RUNモードにします	
RUN	?
1 $\sqrt{1}$ と 1^2 の答	ルートA=1. A^2=1. ?
2 $\sqrt{2}$ と 2^2 の答	ルートA=1.414213562 A^2=4. ?
3 $\sqrt{3}$ と 3^2 の答	ルートA=1.732050808 A^2=9. ?
4 $\sqrt{4}$ と 4^2 の答	ルートA=2. A^2=16. ?
5.23 $\sqrt{5.23}$ と 5.23^2 の答	ルートA=2.286919325 A^2=27. 3529 ?
	表示桁数が多くなり、 A^2 の値が2行にわかれてしま いました。

この〔例題〕③では、

```
40 PRINT 'ルートA='; X; 'A^2='; Y
```

というように、セミコロンによって表示を継続させたので、1行に全部表示されなくなります。

しかし、次のように2つの行に分けることによって、それぞれの答えを1行に表示させることができます。

```
40 PRINT 'ルートA='; X
```

```
43 PRINT 'A^2='; Y
```

小数点以下2桁程度を表示すればよい場合には、プログラムを次のように変更することによって \sqrt{A} 、 A^2 を一度に表示できます。

①小数点第3位以下を切り捨てる場合 INT/整数化 (Integer/インテジャー)

```

10 INPUT A
20 X=SQR A
25 X=INT(100*X)/100 ←
30 Y=A^2
40 PRINT 'ルートA='; X; 'A^2='; Y
45 GOTO 10
50 END

```

(Xを100倍にしたものを整数化し、それを100で割ることを意味しています。Yも同様な処理をすることができます。)

• どのような表示になるか試してみてください。

②小数点第3位以下を四捨五入する場合

```

10 INPUT A
20 X=SQR A
25 X=INT(100*X+0.5)/100 ←
30 Y=A^2
40 PRINT 'ルートA='; X; 'A^2='; Y
45 GOTO 10
50 END

```

(Xを100倍したものに0.5を加え、それを整数化し100で割ることを意味します。Yも同様な処理をすることができます。)

[補足]

25行は $X = (100 * X) \div 100$ としても同じです。

\div (整数の除算) については26ページを参照してください。

• どのような表示になるか試してみてください。

③数値の桁指定をする場合 USING命令

USING命令を用いて桁数を指定し表示させます。〔例題〕③で、Xの値を小数点2桁まで指定するときは、

```
PRINT USING '##.##'
```

↑
符号として1桁必要 小数部は2桁分

とします。小数点第3位以下は切り捨てとなります。

Yの値は符号の1桁と数値の2桁で、合わせて3桁分の指定が必要です。

```

10 INPUT A
20 X=SQR A
30 Y=A^2
40 PRINT USING '##.##'; 'ルートA='; X
43 PRINT USING '###'; 'A^2='; Y
45 GOTO 10
50 END

```

操 作	表 示 部
BASIC RUNモードにします	
RUN	? .
1	ルートA= 1.00 A^2= 1 ?
2	ルートA= 1.41 A^2= 4 ?
3	ルートA= 1.73 A^2= 9 ?
4	ルートA= 2.00 A^2= 16 ?
5.23	ルートA= 2.28 A^2= 27



四捨五入について

USING命令を用いたときでも、少し工夫すると四捨五入した値を表示できます。

正のときは0.005をたし、負のときは0.005を引いてから小数点第3位以下を切り捨てて表示すればよいのです。

つまり $1.234 + 0.005 = 1.239 \xrightarrow{\text{切り捨て}} 1.23$ $-1.234 - 0.005 = -1.239 \xrightarrow{\text{切り捨て}} -1.23$
 $1.236 + 0.005 = 1.241 \xrightarrow{\text{切り捨て}} 1.24$ $-1.236 - 0.005 = -1.241 \xrightarrow{\text{切り捨て}} -1.24$

負のときは、 $-(1.234 + 0.005)$ のようにマイナスでくると、正のときと同じ形になります。正か負の判断のためにSGN (STEP 3 参照) という便利な関数があります。そのため求められたXに対し、 $\text{SGN } X * (\text{ABS } X + 0.005)$ を行います。

↑ 正のときは1、負のときは-1、ゼロのときは0になる関数です。

STEP 3 関数を使うプログラム

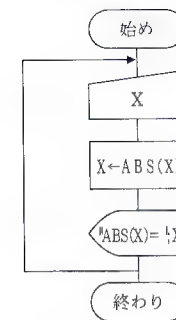
【例題】④

数Xを入力してその絶対値を表示するプログラムを作りなさい。

■解 説

ABS (X) の式でXの絶対値を求めます。[Absolute (アブソリュート：絶対値)]

■フローチャート



【例題】⑤

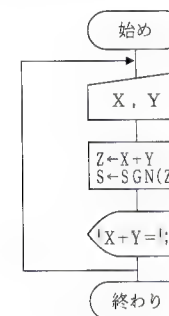
数XとYを入力して、X+Yの値がプラスなら1を、マイナスなら-1を、0なら0を表示するプログラムを作りなさい。

■解 説

符号はSGNで調べることができます。[Sign (シグナム)]

SGN (X) で、X > 0 のとき1、X = 0 のとき0、X < 0 のとき-1が得られます。

■フローチャート



【例題】⑥

1~10の乱数を作り、表示するプログラムを作りなさい。

乱数はRND X で与えられます。[Random (ランダム)]

RND Xにおいて、Xの値により次のような乱数を得ることができます。

● Xが2以上の場合

Xが正の整数のとき： 1以上、X以下の乱数を発生します。

X = 5 のとき…… 1 ~ 5 の間の乱数

Xが小数部を含むとき： 1以上、Xの整数部に1を加えた値以下の乱数を発生します。

X = 5.2 のとき…… 1 ~ 6 の間の乱数

■プログラム例

```

10 INPUT X
20 X = ABS (X)
30 PRINT "ABS (X) = "; X
40 GOTO 10
50 END
  
```

結 果 例	
X	ABS (X)
3	3
5	5
-2	2
-4	4
8	8

■プログラム例

```

10 INPUT X, Y
20 Z = X + Y
30 S = SGN (Z) → ( ) はなくてもよい
40 PRINT "X + Y = "; S
50 GOTO 10
60 END
  
```

結 果 例		
X	Y	S
3	2	1
3	-5	-1
2	-2	0

● Xが負数の場合

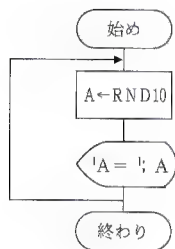
同じ乱数（あるいは乱数列）を発生させるために、初期値を一定にします。

● Xが0以上、2未満の場合

0より大きく、1より小さい値の乱数を発生します。

X=1.42のとき……0～1の間の乱数

■フローチャート



■プログラム例

```
5 WAIT 50
```

```
10 A=RND 10 ←----- この部分の数値を
```

```
20 PRINT 'A = ' ; A
```

```
30 GOTO 10
```

```
40 END
```

2.8 → 小数部あり

-5 → 負 数

0.5 → 1未満

と変えて、乱数の出かたをいろいろ確認してください。

5 WAIT 50 は20行のPRINT命令の表示時間を指定しています。

ただし、一般のパソコンではWAIT指定ができないので次のようにします。

```
25 FOR J =1 TO 500: NEXT J
```

(FOR、NEXTについては121ページ参照)

【例題】⑦

数XとYを入力して X^Y は何桁の数かを求めるプログラムを作りなさい。

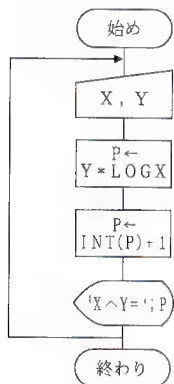
ただし、XとYは正の整数値とします。

■解 説

常用対数を用います。 $P = X^Y$ とおき、両辺の対数をとると、 $\log P = Y \cdot \log X$ になります。

対数の性質から、桁数は $Y \cdot \log X$ の値の整数値+1桁になります。

■フローチャート



■プログラム例

```
10 INPUT X, Y
```

```
20 P=Y*LOG X
```

```
30 P=INT(P)+1
```

```
40 PRINT X ; '^' ; Y ; '=' ; P ; 'ケタ'
```

```
50 GOTO 10
```

```
60 END
```

メモリ内容

変 数	内 容
X	入力値
Y	入力値
P	桁 数

【例題】⑧

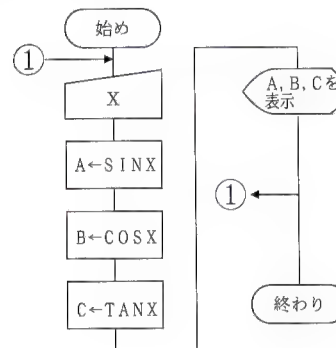
θ に適当な数を入れ sin, cos, tan の値を求めるプログラムを作りなさい。

($\theta = 30, 45, 60$ と入力しなさい。)

■解 説

三角関数キーを用います。角度単位を度(DEG)に指定します。

■フローチャート



■プログラム例

```
5 DEGREE
```

```
10 INPUT X
```

```
20 USING
```

```
30 A=SIN X: B=cos X: C=TAN X
```

```
40 PRINT 'SIN'; X; '=';
```

```
; USING '###.####'; A
```

```
50 USING
```

```
60 PRINT 'COS'; X; '=';
```

```
; USING '###.####'; B
```

```
70 USING
```

```
80 PRINT 'TAN'; X; '=';
```

```
; USING '###.####'; C
```

```
90 GOTO 10
```

```
100 END
```

~~~~~ 線で示す内容がある場合と、ない場合の表示のしかたの違いを確認してください。



## USINGについて

【例題】⑧の20行を20 USING '###'と変更し、実行してみてください。

「SIN 30=」と30に小数点がつかない形で表示されます。(33ページ参照)

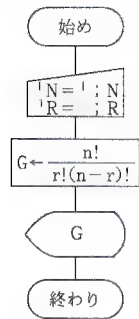
## 【例題】⑨

組合せ計算  $nCr = \frac{n!}{r!(n-r)!}$  の値を求めるプログラムを作りなさい。

## ■解 説

組合せ(NCR)を用いれば簡単に計算できますが、ここでは階乗(FACT)を用いてプログラムを作ってみます。

## ■フローチャート



## ■プログラム例

```

10 INPUT 'N='; N, 'R='; R
20 G=FACT N / (FACT R*FACT (N-R))
30 PRINT G
40 END

```

(注) 20行は  $G = \text{NCR}(N, R)$  でもよい。

| メモリ内容 |                 |
|-------|-----------------|
| 変数    | 内容              |
| N     | 入力値             |
| R     | 入力値             |
| G     | $n! / r!(n-r)!$ |

| 結果例 |   |     |
|-----|---|-----|
| n   | r | nCr |
| 5   | 3 | 10  |
| 10  | 4 | 210 |

## 演習問題 2

を行ってください。

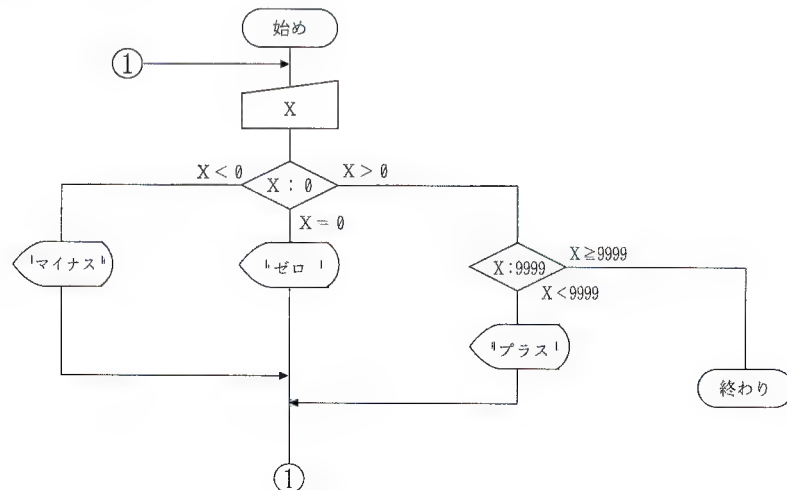
## STEP 4 IF~THEN~ELSE

## 【例題】⑩

数Xを入力し、正の数なら「プラス デス」  
負の数なら「マイナス デス」  
0 なら「ゼロ デス」

と表示するプログラムを作りなさい。ただし、データが9999以上のときは終わるプログラムを作りなさい。

## ■フローチャート



## ■プログラム例

```

10 INPUT 'X='; X
20 IF X < 0 THEN 70
30 IF X = 0 THEN 90
40 IF X >= 9999 THEN 110
50 PRINT 'プラス デス'
60 GOTO 10
70 PRINT 'マイナス デス'
80 GOTO 10
90 PRINT 'ゼロ デス'
100 GOTO 10
110 END

```

(注) ELSEを省略した例です。

## ■解説

- ① IF~THEN~ELSEは、IF以降の条件が成立するときはTHEN以降の命令に従い、条件が成立しないときはELSE以降の命令に従います。ELSEを省略した場合は、条件が成立しないとき、次の行に進みます。  
THEN (またはELSE) 行番号 ..... その行番号へ行きます。  
THEN (またはELSE) ステートメント ..... そのステートメントを実行した後、次の行番号へ行きます。
- ② IF~THEN 行番号はIF~GOTO 行番号と同じ命令です。  
「THEN GOTO 行番号」という命令文は、「THEN 行番号」または「GOTO 行番号」という形に省略できます。  
「ELSE GOTO 行番号」は、「ELSE 行番号」という形に省略できます。(「GOTO 行番号」にはできません。)  
例) 10 IF A=5 THEN B=0 ELSE 200  
Aが5ならばBを0にして次の行に移り、Aが5以外ならば200行に移ります。

| 条件式         | 判断内容                        |
|-------------|-----------------------------|
| $○○ = ××$   | 等しいかどうか判断 (○○は××に等しいか?)     |
| $○○ > ××$   | 大きいかどうか判断 (○○は××より大きいか?)    |
| $○○ ≥ ××$   | 以上かどうか判断 (○○は××以上か?)        |
| $○○ < ××$   | 小さいかどうか判断 (○○は××より小さいか?)    |
| $○○ ≤ ××$   | 以下かどうか判断 (○○は××以下か?)        |
| $○○ < > ××$ | 等しくないかどうか判断 (○○と××は等しくないか?) |

## 【例題】⑪

二次方程式  $ax^2 + bx + c = 0$  ( $a \neq 0$ ) の解の種類を判別して解を求めるプログラムを作りなさい。ただし、2実根のときは「2ジコン」、重根のときは「ジュウコン」、虚根のときは「キョコン」と表示させてそれぞれの解を求めなさい。

## ■解説

一般式  $ax^2 + bx + c = 0$  ( $a \neq 0$ ) の解の公式は  $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

$D = b^2 - 4ac$  とすると

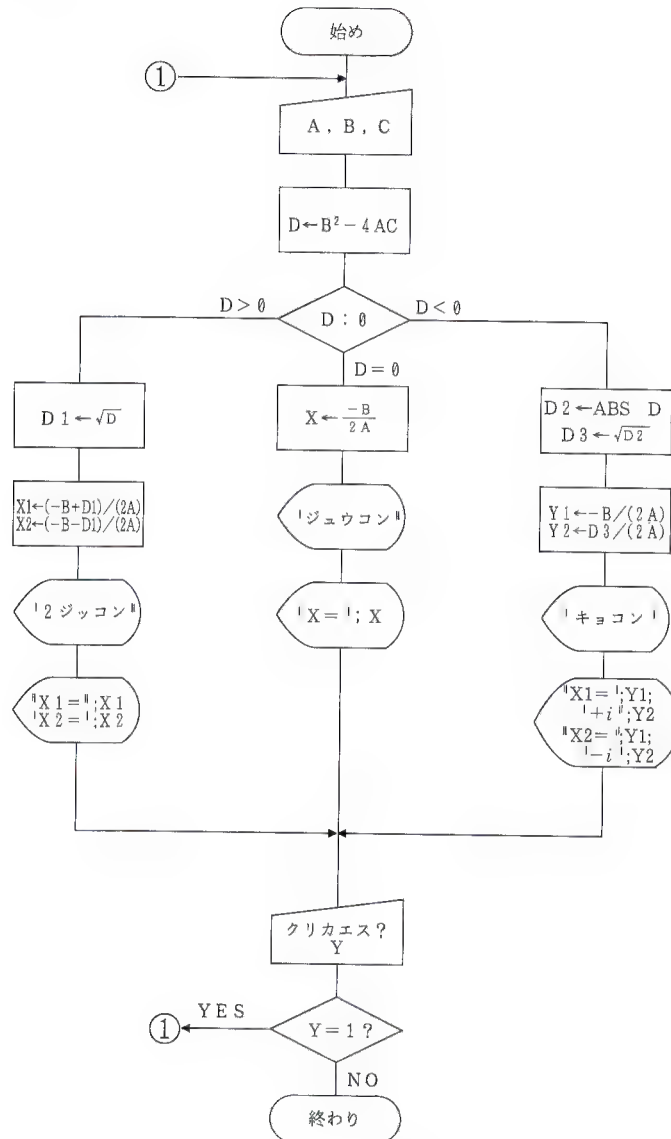
$D > 0$  ..... 2実根 ( $x_1 = \frac{-b + \sqrt{D}}{2a}$ ,  $x_2 = \frac{-b - \sqrt{D}}{2a}$ )

$D = 0$  ..... 重根  $x = \frac{-b}{2a}$

$D < 0$  ..... 虚根 ( $x_1 = \frac{-b + \sqrt{D}i}{2a}$ ,  $x_2 = \frac{-b - \sqrt{D}i}{2a}$ )



## ■フローチャート



## 虚数について

√の中が負になると、一般にコンピュータではエラーになり計算できません。  
 そのため、Dの符号を判断し、Dの絶対値を考えて計算する必要があります。  
 数学では、 $i^2 = -1$  になる記号を考え出し、 $\sqrt{-D} = \sqrt{D}i$  と表します。 $i$  を虚数単位と呼びます。  
 なお、電気関係では電流を  $i$  と表現しますので、虚数単位を  $j$  と表すこともあります。

## ■プログラム例

```

10 INPUT "A="; A, "B="; B, "C="; C
20 D=B^2-4*A*C
30 IF D<0 THEN 160
40 IF D=0 THEN 120
50 D1=SQR D
60 X1=(-B+D1)/(2*A)
70 X2=(-B-D1)/(2*A)
80 PRINT "2 ジュウコン"
90 PRINT "X1="; X1
100 PRINT "X2="; X2
110 GOTO 210
120 X=-B/(2*A)
130 PRINT "1 ジュウコン"
140 PRINT "X="; X
150 GOTO 210
160 D2=ABS D: D3=SQR D2
170 Y1=-B/(2*A): Y2=D3/(2*A)
180 PRINT "1 キョウコン"
190 PRINT "X1="; Y1; " + i "; Y2
200 PRINT "X2="; Y1; " - i "; Y2
210 INPUT "クリカエス?-(YES=1)"; Y
220 IF Y=1 THEN 10
230 END
  
```

| メモリ内容 |                 |
|-------|-----------------|
| 変数    | 内容              |
| A     | 2次係数            |
| B     | 1次係数            |
| C     | 定数              |
| D     | $B^2 - 4AC$ 判別式 |
| D1    | $\sqrt{D}$      |
| D2    | ABS D           |
| D3    | $\sqrt{D2}$     |
| X     | $-B/2A$         |
| X1    | $(-B + D1)/2A$  |
| X2    | $(-B - D1)/2A$  |
| Y1    | $-B/2A$         |
| Y2    | $D3/2A$         |
| Y     | 繰り返し判断用         |

下表の数値を入れて結果を求めてください。

| A | B            | C               | 結 果                                                     |
|---|--------------|-----------------|---------------------------------------------------------|
| 1 | 5            | 6               | 2 ジュウコン X1=-2 X2=-3                                     |
| 1 | 2            | 1               | ジュウコン X=-1                                              |
| 5 | -10          | 25              | キョウコン X1=1+i2 X2=1-i2                                   |
| 1 | 1            | $-3 + \sqrt{3}$ | 2 ジュウコン X1=7.320508075×10 <sup>-1</sup> X2=-1.732050808 |
| 1 | -3           | 6               | キョウコン X1=1.5+i1.936491673 X2=1.5-i1.936491673           |
| 2 | $2\sqrt{10}$ | 5               | ジュウコン X=-1.58113883                                     |

## 演習問題

3

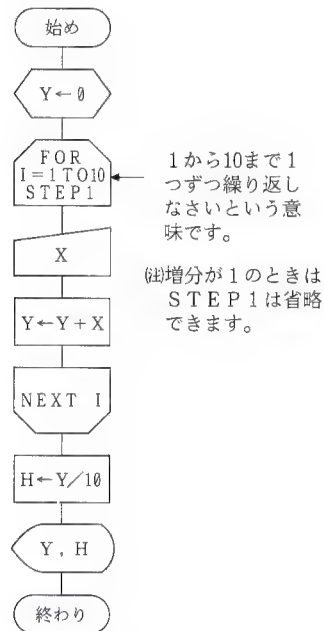
を行ってください。

## STEP 5 FOR~TO~STEP、NEXT

【例題】⑫

数値を10個入力して合計と平均を求めるプログラムを作りなさい。

## ■フローチャート



## ■プログラム例

```

10 Y = 0
20 FOR I = 1 TO 10 STEP 1
30 INPUT 'X='; X
40 Y = Y + X
50 NEXT I
60 H = Y / 10
70 PRINT 'コウケイ='; Y
80 PRINT 'ヘイキン='; H
90 END

```

対応しています

| メモリ内容 |                               |
|-------|-------------------------------|
| 変数    | 内容                            |
| Y     | 初期値設定<br>Y = 0<br>累計<br>Y + X |
| X     | 入力値                           |
| H     | 平均<br>Y / 10                  |

## ■解説

- ①行番号10のY = 0は、変数Yの中には何も数値が入っていないという状態を作っています。この状態を最初におかないと、行番号40のところで誤った値を加算することになります。
- ②行番号20のSTEP 1はこの値が1の場合に限って省略できます。  
FOR IのIとNEXT IのIは同じ文字(変数)にしなければなりません。  
ただし、NEXT IのIは省略できます。たとえば、FOR J = 1 TO 10とすれば、NEXT Jとするか、NEXTということになります。  
しかし、慣れるまではNEXTの後の文字を省略しないほうがよいでしょう。  
プログラムを見直すときは、NEXTの後の文字があるほうが便利です。
- ③行番号40のY = Y + Xは累計を求めています。つまり、Yに新しいXの値を加えています。
- ④行番号60のH = Y / 10は平均を求めています。

## ■累計の数値例

| Iの値         | 1     | 2     | 3     | 4     | 5      | 6      | 7      | 8      | 9      | 10      |
|-------------|-------|-------|-------|-------|--------|--------|--------|--------|--------|---------|
| Y + X       | 0 + 1 | 1 + 2 | 3 + 3 | 6 + 4 | 10 + 5 | 15 + 6 | 21 + 7 | 28 + 8 | 36 + 9 | 45 + 10 |
| Y = Y + Xの値 | 1     | 3     | 6     | 10    | 15     | 21     | 28     | 36     | 45     | 55      |

- 上の表のようにYの値はIの値の変化とともにそれぞれ累計されていきます。

FOR~NEXT文を使用しないで合計を求めるプログラム

【例題】⑫で、FOR~NEXT文を使用しないで累計を求める方法

## ■プログラム例

```

10 A = 0
20 Y = 0
30 INPUT 'X='; X
40 A = A + 1
50 Y = Y + X
60 IF A < 10 THEN 30
70 H = Y / 10
80 PRINT 'コウケイ='; Y
90 PRINT 'ヘイキン='; H
100 END

```

## ■解説

- ①このプログラムは30行と60行の間を条件が成立するまで、繰り返しています。(繰り返しループを形成します)
- ②40行のA = A + 1は左辺のAが10になるまで60行のIF文で判断され、このループで10回のたし算を行います。  
Aをカウンタとして利用しています。
- ③50行のY = Y + Xで、Xの値が左辺のYの変数に累計されます。Aが10になった時点で累計完了ということになります。

## 研究演習問題

データが10個ではなく、N個の場合(Nは任意の正の整数)の合計と平均を求めるプログラムを考えてください。

## 「FOR~NEXT」文の二重ループ

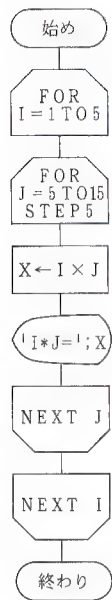
【例題】⑬

|                                                                                                                                        |                             |
|----------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| $1 \times 5, 1 \times 10, 1 \times 15$<br>$2 \times 5, 2 \times 10, 2 \times 15$<br>$\vdots$<br>$5 \times 5, 5 \times 10, 5 \times 15$ | をそれぞれ表示し、答也表示するプログラムを作りなさい。 |
|----------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|

## ■解説①

- 1~5の1つ刻みのループ(STEP 1)と  
5~15の5つ刻みのループ(STEP 5)  
がありますので、FOR~NEXTの二重ループとなります。

## ■フローチャート



## ■プログラム例

```

10 FOR I=1 TO 5
20 FOR J=5 TO 15 STEP 5
30 X ← I * J
40 PRINT I ; ' * ' ; J ; ' = ' ; X
45 FOR K=1 TO 500 : NEXT K
50 NEXT J
60 NEXT I
70 END

```

## ■解 説②

行番号20と50が対応し、行番号10と60が対応しています。IとJの値について、ループ内での数値の動きは

I = 1 → J = 5, J = 10, J = 15となってI = 2に移ります。  
 I = 2 → J = 5, J = 10, J = 15となってI = 3に移ります。  
 I = 3 → J = 5, J = 10, J = 15となってI = 4に移ります。  
 I = 4 → J = 5, J = 10, J = 15となってI = 5に移ります。  
 I = 5 → J = 5, J = 10, J = 15

となり、これでプログラムは終了となります。

## ■FOR~NEXTについて

〈例1〉このプログラム例が正しい使いかたです。ループ①はループ②の中に完全に入っていないければなりません。使用できる最大のループ数については327ページのBASICの各命令の説明を参照してください。

```

10 FOR I=1 TO 5
:
40 FOR J=1 TO 10
:
80 NEXT J
:
100 NEXT I

```

```

〈例2〉10 FOR I=1 TO 5
:
40 FOR J=1 TO 10
:
80 NEXT I
:
100 NEXT J

```

このような、プログラムはERROR52となります。ループ①とループ②が交差することは文法上許されません。

〈例3〉40 GOTO 80

外からFOR~NEXTループ内にとび込むことはできません。

```

:
60 FOR B=-10 TO 12 STEP 2
:
80 D=B^2-4*A
:
100 NEXT B

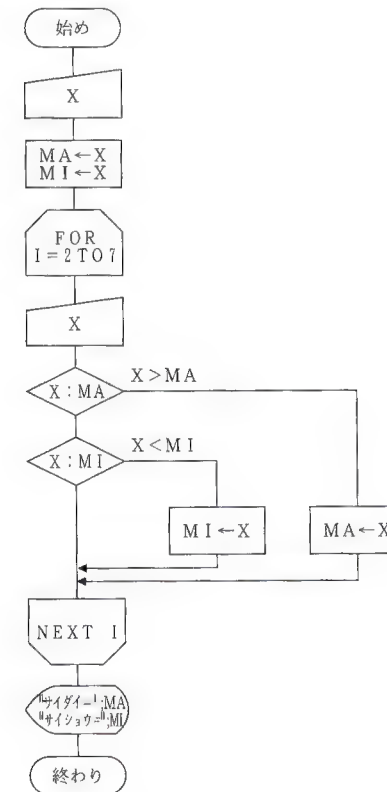
```

ループ

## 【例題】⑭

7つのデータがあります。15、21、36、81、9、16、10をそれぞれ入力し、最大値と最小値を求めるプログラムを作らないさい。

## ■フローチャート



## ■プログラム例

```

10 INPUT "X=" ; X
20 MA=X : MI=X
30 FOR I=2 TO 7
40 INPUT "X=" ; X
50 IF X>MA THEN MA=X : GOT
  O 70
60 IF X<MI THEN MI=X
70 NEXT I
80 PRINT "サイタ" I=" ; MA ; "サイショ
  ウ=" ; MI
90 END

```

| メモリ内容 |        |
|-------|--------|
| 変数    | 内容     |
| MA    | 最大値    |
| MI    | 最小値    |
| X     | データ入力値 |

## ■解 説

- ①行番号10で最初のデータを入力します。
- ②行番号20のMA=X : MI=Xで、最初の入力データを仮の最大値、最小値にします。
- ③行番号30~70で、データを1つずつ入力していきます。
- ④行番号50~60で、最大値、最小値の比較を行っています。

下の表で最大値、最小値を比較している内容を細かく分析してみます。

| Xの値   | 最大値の比較<br>$X > MA$ | 最大値    | 最小値の比較<br>$X < MI$ | 最小値   |
|-------|--------------------|--------|--------------------|-------|
| 15のとき | —                  | 15となる  | —                  | 15となる |
| 21のとき | $21 > 15$          | 21に変わる | $21 < 15$          | 15のまま |
| 36のとき | $36 > 21$          | 36に変わる | $36 < 15$          | 15のまま |
| 81のとき | $81 > 36$          | 81に変わる | $81 < 15$          | 15のまま |
| 9のとき  | $81 > 9$           | 81のまま  | $9 < 15$           | 9に変わる |
| 16のとき | $16 > 81$          | 81のまま  | $16 < 9$           | 9のまま  |
| 10のとき | $10 > 81$          | 81のまま  | $10 < 9$           | 9のまま  |

↑  
最終結果・最大値

↑  
最終結果・最小値

## 演習問題

4

を行ってください。

## STEP 6 REM、READ、DATA、RESTORE

【例題】⑬

55、70、80、65、50を読んで、平均を求めるプログラムを作りなさい。ただし、ヘイキンという注釈文をプログラムリストの最初につけなさい。

### ■解説

①注釈文をつけるときは、REM文を用います。(REMまたは‘)

REMは、注釈の意味でプログラムの実行には関係ありません。しかし、プログラムリストをみるとどのようなプログラムなのかがわかります。

また、長いプログラムなどにおいては、演算処理の項目ごとにREM文で注釈をつけておくと、後でプログラムリストを検討するときにとてもわかりやすくなります。

②REM文は、任意の行番号をつけることができます。

③READ文は、DATA文と対になって変数にデータを割りあてます。

④対応するデータがなくなると、ERROR53が発生します。

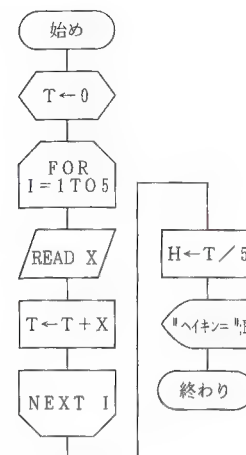
### ■プログラム例

```

10 REM ヘイキン
20 T = 0
30 FOR I = 1 TO 5
40 READ X
50 T = T + X : REM コウケイ
60 NEXT I
70 H = T / 5
80 PRINT "ヘイキンは"; H
90 END
100 DATA 55, 70, 80, 65, 50

```

### ■フローチャート



### ■結果

ヘイキンは 64.

### ■プログラムの解説

10 ヘイキンという注釈文になります。10'ヘイキンとしてもかまいません。

20 Tの変数を0(ゼロ)にします。ここに、CLEAR命令を用いてもかまいません。

30 30行から60行のFOR~NEXTループです。Iの値が1から始めて、5を超えるまで繰り返します。

40 READ文です。

READ Xは、100行のDATA文の最初のデータ55を読み込みます。

50 累計を求めています。このときのTは、 $T = T + X$ 。20行の0(ゼロ)と55(55)が加算されます。

60 NEXT Iで30行に戻ります。このとき、すでに、 $I = 2$ となっています。

40行のREAD Xは、再度100行の次のDATA文の70を読み込みます。

50行は  $T = T + X$  ← DATA 70  
125 ← I=1のときのTの合計 55

このように、残りのデータ80、65、50も同じように読み込み、たしていきます。

70  $H = T / 5$   
320が入ります(5つのデータの合計です)

したがって、変数Hには、64という数字が入ります。

80 Hの値を表示します。

90 プログラムの実行が終了します。

100 DATA文です。

### ★★★★★

行番号50の:REM コウケイ は、注釈文です。:(コロン)は継続行を意味しますから、このような形でREM文を使うこともできます。REM文で注釈をつけておくと、後でプログラムリストを解析するときにとてもわかりやすくなります。

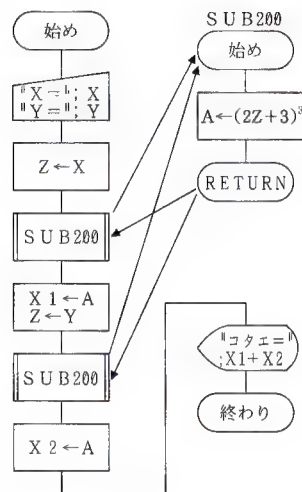




## 【例題】⑩

数XとYを入力し、 $(2X+3)^3 + (2Y+3)^3$ を求めるプログラムを作りなさい。

## ■フローチャート



| 結果例 |    |       |
|-----|----|-------|
| X   | Y  | コタエ   |
| 1   | 1  | 250   |
| 1   | 2  | 468   |
| 5   | 10 | 14364 |

## ■解説

- ①  $(2X+3)^3$  と  $(2Y+3)^3$  はXとYをZに置き換えると同じ形となります。この計算式をサブプログラムにしておくと、何度も書く必要がなくなり簡単な形のプログラムにできます。
- ② サブプログラムで  $(2X+3)^3$  を求めるために  $Z = X$  として、Xの値をZに代入しておきます。(ここが重要です)
- ③ サブプログラムの計算結果は  $X1 = A$  としてX1に代入します。
- ④ 同じように  $Z = Y$  として、Yの値をZに代入しておきます。
- ⑤ サブプログラムの計算結果は  $X2 = A$  としてX2に代入しておきます。

## ■プログラム例

```

10 INPUT "X=" ; X
20 INPUT "Y=" ; Y
30 Z=X
40 GOSUB 200
50 X1=A
60 Z=Y
70 GOSUB 200
80 X2=A
90 PRINT "コタエ=" ; X1+X2
100 END
200 A=(2*Z+3)^3
210 RETURN

```

## 演習問題 6

を行ってください。

## STEP 8 配列DIM (ディメンジョン)

## 【例題】⑪

5個の数 25、18、23、17、9を読み込んで、読み込んだ順に配列変数Bに代入し、次に5個の数の合計を求めて表示するプログラムをDIM文を用いて作りなさい。

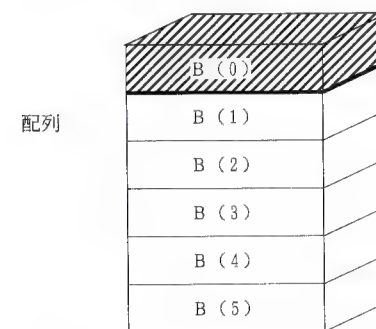
## ■フローチャート



## 1次元配列

DIM B(5) の意味  
添字：これは配列の大きさです。  
この数字は必ず整数を用います。  
数値配列変数  
英文字を1文字または2文字使用できます。また、英文字と数字の組み合わせも使用できます。  
配列 (ディメンジョン) を意味します。  
変数を下図のように、データをしまひ込む (出し入れをする) 箱と考えたとき、DIM文は箱の数を決めて用意するステートメントと考えます。

DIM B(5) は



- B(0)からB(5)までB(0)を含めて、6個の箱が用意されます。
- この例題のプログラム例では斜線の部分は未使用ということになります。



## 配列の添字について

配列の大きさを指定する数値を添字そえじといいます。添字が1つのものを1次元配列 (例: B(5))、添字が2つのものを2次元配列 (例: B(5, 2)) といいます。添字がn個あれば、n次元配列ということになります。本機は2次元配列までできます。  
また、コンピュータでは一般に、B(0)やB(0, 0)のように0から始まる変数が確保されるので、人間の感覚とは若干ずれます。

## ■プログラム例

```

10 DIM B(5)
20 FOR I=1 TO 5
30 PRINT "B( "; I; ") = ";
40 INPUT B(I)
50 NEXT I
60 S=0
70 FOR I=1 TO 5
80 S=S+B(I)
90 NEXT I
100 PRINT "ゴウケイ="; S
110 END

```

| 結 果 例   |
|---------|
| B(1)=25 |
| B(2)=18 |
| B(3)=23 |
| B(4)=17 |
| B(5)=9  |
| ゴウケイ 92 |

このようにB(I)のIには、順次1、2、3、4、5という数字が代入されていきます。

## ■プログラムの解説

10 配列を指定します。

20 5個（入力する値が5個）の数値が入るループを作ります。

30 入力する順番を表示させます。最後の；は、40行で入力される内容を続けて表示させるために入れています。

40 数値を入力します。30行で表示した内容の後に入ります。

50 5個の値を入力するまで20行へ行きます。

60 合計を求める前に、今の合計値を0に指定します。（まだ計算をしていないためです。）  
ここでCLEAR命令を使うと、せっかく入力した5個の値が消えてしまいます。

70 B(I)の箱の中に入れた5個の数値を1から5まで取り出すループを作ります。

80 1つつ取り出したら、それぞれ累計します。

90 5個の値をB(I)から取り出すまで70行へ行きます。

100 5個の値を全て取り出した後、累計した結果を表示します。

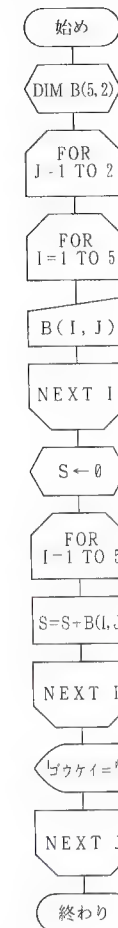
110 プログラムの実行が終了します。

## 〔例 題〕②

右の図のように5個の数が2列あります。  
列ごとの合計を求めて表示するプログラムを作りなさい。

|     |    |    |
|-----|----|----|
|     | 25 | 38 |
|     | 18 | 46 |
|     | 23 | 92 |
|     | 17 | 73 |
|     | 9  | 65 |
| 合 計 |    |    |

## ■フローチャート



## 2次元配列

DIM B(5, 2)  
(行, 列)

5行, 2列の配列を意味します。

DIM B(5, 2)の2次元配列では下図のような箱が用意されます。

|         |               |               |
|---------|---------------|---------------|
| B(0, 0) | B(0, 1)       | B(0, 2)       |
| B(1, 0) | B(1, 1)<br>25 | B(1, 2)<br>38 |
| B(2, 0) | B(2, 1)<br>18 | B(2, 2)<br>46 |
| B(3, 0) | B(3, 1)<br>23 | B(3, 2)<br>92 |
| B(4, 0) | B(4, 1)<br>17 | B(4, 2)<br>73 |
| B(5, 0) | B(5, 1)<br>9  | B(5, 2)<br>65 |

例題②では斜線の部分は使用していません。

## ■プログラム例

```

10 DIM B(5, 2)
20 FOR J=1 TO 2
30 FOR I=1 TO 5
40 INPUT B(I, J)
50 NEXT I
60 S=0
70 FOR I=1 TO 5
80 S=S+B(I, J)
90 NEXT I
100 PRINT "ゴウケイ="; S
110 NEXT J
120 END

```

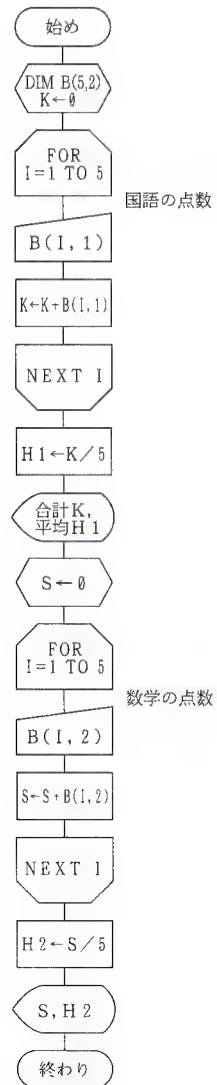
| メモリー内容 |                                 |
|--------|---------------------------------|
| 変数     | 内 容                             |
| B      | B(5, 2) 2次元配列<br>B(I, J)        |
| I      | I = 1 ~ 5                       |
| J      | J = 1 ~ 2                       |
| S      | 合 計<br>S = 0<br>S = S + B(I, J) |

## 【例題】⑪

下表のような5人の生徒の成績があります。科目ごとの合計と平均点を計算するプログラムを作りなさい。

| 番号 | 1  | 2  | 3  | 4  | 5  |
|----|----|----|----|----|----|
| 国語 | 58 | 45 | 69 | 87 | 75 |
| 数学 | 45 | 25 | 78 | 82 | 65 |

## ■フローチャート



## ■プログラム例

```

10 DIM B(5,2)
20 K=0
30 FOR I=1 TO 5
40 INPUT "コクコ テンスウ="; B(I,1)
50 K=K+B(I,1)
60 NEXT I
70 H1=K/5
80 PRINT "コ ウケイ="; K; "ヘイキン="; H1
90 S=0
100 FOR I=1 TO 5
110 INPUT "スウカ クーテンスウ="; B(I,2)
120 S=S+B(I,2)
130 NEXT I
140 H2=S/5
150 PRINT "コ ウケイ="; S; "ヘイキン="; H2
160 END

```

| メモリ内容 |                       |  |
|-------|-----------------------|--|
| 変数    | 内 容                   |  |
| B( )  | 国語の点数 数学の点数           |  |
| H1    | H1=K/5 国語の平均          |  |
| H2    | H2=S/5 数学の平均          |  |
| K     | K=0 K=K+B(I,1) 国語の合計点 |  |
| S     | S=0 S=S+B(I,2) 数学の合計点 |  |
| I     | I=1~5                 |  |

## ■プログラムの解説

10 2次元配列、5行・2列を指定します。  
 20 国語の合計を求めるためにK=0としておきます。  
 30~60 Iを1から5へと順に進めて、国語の点数を順次入力します。50行で合計を求めるために累計しています。  
 70 5人分の国語の平均点を求めます。  
 80 合計と平均値を表示します。  
 90~150 今度は数学の計算を行います。国語の場合とほぼ同じです。  
 ただ、国語はB(I,1)の変数を使いましたが、数学ではB(I,2)の変数を使います。

## 【例題】⑫

下のような数値の表があります。

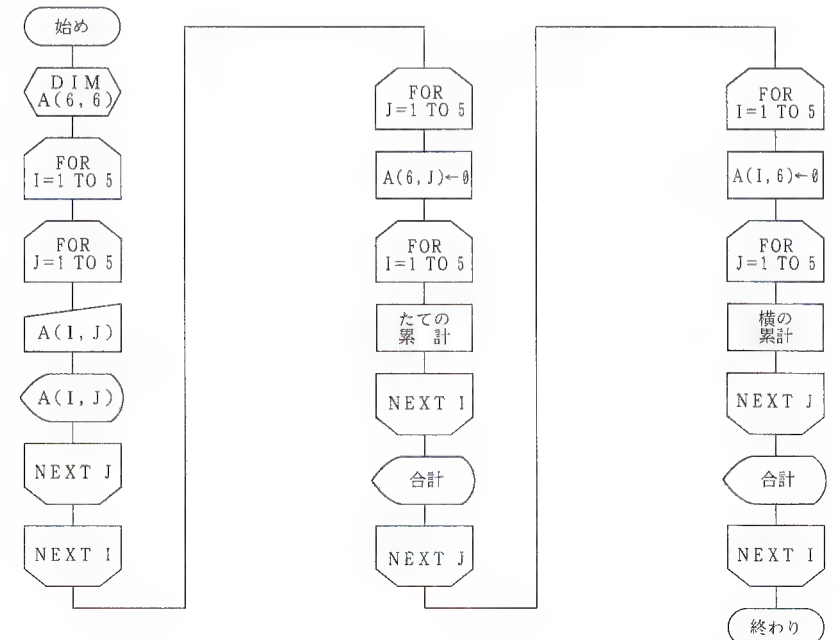
数値を、1、2、3、4、5

11、12、13、14、15

と順次入力していき、たて、横の合計を求めるプログラムを作りなさい。

|       |    |    |    |    | 横の合計 |
|-------|----|----|----|----|------|
| 1     | 2  | 3  | 4  | 5  |      |
| 11    | 12 | 13 | 14 | 15 |      |
| 21    | 22 | 23 | 24 | 25 |      |
| 31    | 32 | 33 | 34 | 35 |      |
| 41    | 42 | 43 | 44 | 45 |      |
| たての合計 |    |    |    |    |      |

## ■フローチャート





## ■プログラム例

```

10 DIM A (6, 6)
20 FOR I=1 TO 5
30 FOR J=1 TO 5
40 INPUT A (I, J)
50 PRINT 'A ( ' ; I ; ', ' ; J ; ') = ' ; A (I, J)
60 NEXT J
70 NEXT I
80 FOR J=1 TO 5
90 A (6, J) = 0
100 FOR I=1 TO 5
110 A (6, J) = A (6, J) + A (I, J)
120 NEXT I
130 PRINT 'A (6, ' ; J ; ') = ' ; A (6, J)
140 NEXT J
150 FOR I=1 TO 5
160 A (I, 6) = 0
170 FOR J=1 TO 5
180 A (I, 6) = A (I, 6) + A (I, J)
190 NEXT J
200 PRINT 'A ( ' ; I ; ', ' ; 6) = ' ; A (I, 6)
210 NEXT I
220 END

```

| メモリ内容    |                |
|----------|----------------|
| 変数       | 内容             |
| A ( )    | A (6, 6) 2次元配列 |
| A (I, J) | 入力値            |
| A (6, J) | たての合計          |
| A (I, 6) | 横の合計           |
| I        | I = 1 ~ 5      |
| J        | J = 1 ~ 5      |

## ■プログラムの解説

10 2次元配列を指定します。  
 20~70 データを入力し表示するループです。  
 90 A (6, J) = 0 はたての合計です。計算前なので0を代入します。  
 160 A (I, 6) = 0 は横の合計です。計算前なので0を代入します。  
 40行、50行目について、実行例で説明をします。

```

RUN  ⏎ ?
1    ⏎ A (1., 1.) 1.
      ?

```

```

2    ⏎ A (1., 2.) = 2.
      ?
3    ⏎ A (1., 3.) = 3.
      ?
4    ⏎ A (1., 4.) = 4.
      ?
5    ⏎ A (1., 5.) = 5.
      ?
11   ⏎ A (2., 1.) = 1 1.
      ?
12   ⏎ A (2., 2.) = 1 2.
      ?

```

このように、40行と50行目は、何行何列目にどのような数値が入ったかを確認できるようにしたものです。  
 110行目のたての累計計算は、下に示すような形でコンピュータは計算しています。

| A(6, J) = A(6, J) + A(I, J) |       | J = 1 のとき     | J = 2 のとき     | J = 3 のとき     | J = 4 のとき     | J = 5 のとき    |
|-----------------------------|-------|---------------|---------------|---------------|---------------|--------------|
| たての<br>累計計<br>算の途<br>中結果    | 初め    | A(6, 1)=0     | →A(6, 2)=0    | →A(6, 3)=0    | →A(6, 4)=0    | →A(6, 5)=0   |
|                             | I は 1 | A(6, 1)=1     | A(6, 2)=2     | A(6, 3)=3     | A(6, 4)=4     | A(6, 5)=5    |
|                             | I は 2 | A(6, 1)=12    | A(6, 2)=14    | A(6, 3)=16    | A(6, 4)=18    | A(6, 5)=20   |
|                             | I は 3 | A(6, 1)=33    | A(6, 2)=36    | A(6, 3)=39    | A(6, 4)=42    | A(6, 5)=45   |
|                             | I は 4 | A(6, 1)=64    | A(6, 2)=68    | A(6, 3)=72    | A(6, 4)=76    | A(6, 5)=80 ⑤ |
| 最終結果<br>(合計)                | I は 5 | A(6, 1)=105 ① | A(6, 2)=110 ② | A(6, 3)=115 ③ | A(6, 4)=120 ④ | A(6, 5)=125  |

A(6, J) = A(6, J) + A(I, J) の算術式の I と J では、繰り返しループ (FOR~NEXT文) によって上の表のような順序 ①→②→③→④→⑤ で計算が行われています。

180行目の横の累計計算も同様に、下に示すような形でコンピュータは計算しています。

| A(I, 6) = A(I, 6) + A(I, J) |       | I = 1 のとき    | I = 2 のとき    | I = 3 のとき    | I = 4 のとき     | I = 5 のとき     |
|-----------------------------|-------|--------------|--------------|--------------|---------------|---------------|
| 横の累<br>計計算<br>の途中<br>結果     | 初め    | A(1, 6)=0    | →A(2, 6)=0   | →A(3, 6)=0   | →A(4, 6)=0    | →A(5, 6)=0    |
|                             | J は 1 | A(1, 6)=1    | A(2, 6)=11   | A(3, 6)=21   | A(4, 6)=31    | A(5, 6)=41    |
|                             | J は 2 | A(1, 6)=3    | A(2, 6)=23   | A(3, 6)=43   | A(4, 6)=63    | A(5, 6)=83    |
|                             | J は 3 | A(1, 6)=6    | A(2, 6)=36   | A(3, 6)=66   | A(4, 6)=96    | A(5, 6)=126   |
|                             | J は 4 | A(1, 6)=10 ① | A(2, 6)=50 ② | A(3, 6)=90 ③ | A(4, 6)=130 ④ | A(5, 6)=170 ⑤ |
| 最終結果<br>(合計)                | J は 5 | A(1, 6)=15   | A(2, 6)=65   | A(3, 6)=115  | A(4, 6)=165   | A(5, 6)=215   |

## 演習問題

7

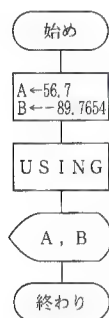
を行ってください。

## STEP 9 USING (ユージング) PRINT USING

### 【例題】23

A=56.7、B=-89.7654の2つの数値において、それぞれ小数点以下第3位で桁をそろえて表示するプログラムを作りなさい。

#### ■フローチャート



#### ■プログラム例

```

10 A=56.7 : B=-89.7654
20 USING '#####.###'
30 PRINT A ; B
40 END
  
```

20行で桁数の指定をします。

#### ■実行手順

RUN モード  
RUN 

表示例

56.700 -89.765

### 【例題】24

B=-10、C=50.8803の2つの数値において、文字と数値をそれぞれ表示するようにプログラムを考えなさい。小数点のあるCについては小数点以下2桁まで表示するものとします。

#### ■フローチャート (省略)

#### ■解説

USINGで数値を指定するときは#マークを使いますが、文字を指定するときは&マークを使います。文字の数だけ&マークを続けて指定します。

#### ■プログラム例

```

10 B=-10 : C=50.8803
20 PRINT USING '&&&###'; B; 'C='; USING '###.##'
30 END
  
```

・&&&は'B='と'C='で2回使用しています。

実行表示例 B = -10 C = 50.88

(注)

桁数指定を解除したいときは、解除したい行にUSINGのみを指定します。USINGを指定した行以降は桁数指定が解除されます。

## STEP 10 MID\$ (ミッド・ドル) .....中間の文字を取り出す LEN (レングス) .....文字列の文字を数える VAL (バリュー) .....文字から数字へ変換する

### 【例題】25

2進数を10進数に変換するプログラムを作りなさい。

#### ■解説

今までは数値だけを扱ってきましたが、これから文字列に関する練習をやってみましょう。コンピュータは数字か“文字”かをはっきり指示してやらなければ動きませんが、逆に言う人間にはできないこともやってくれます。

たとえば、2進数の10010110は10進数ではいくつになるでしょうか。

1 0 0 1 0 1 1 0  
1        11        11  
2<sup>7</sup> + 2<sup>4</sup> + 2<sup>2</sup> + 1<sup>1</sup>  
128 + 16 + 4 + 2 = 150  
このような演算を上から実行し、加算することになります。

#### ■プログラム例

```

10 : 'A' : CLEAR :
    DIM B$(0)
20 : INPUT B$(0)
30 : L=LEN B$(0)
40 : N=1
50 : C$=MID$(B$(0),
    N, 1)
60 : E=VAL C$
70 : D=E*2^(L-1)+D
80 : N=N+1 : L=L-1
90 : IF L=0 THEN
    110
100 : GOTO 50
110 : PRINT D
120 : GOTO 10
  
```

10 最初にDIM B\$(0)と配列宣言をして、代入する文字数を16に設定します。つまり、入力できる2進数は最大16桁までになります。B\$だけにすると、最大7文字しか入力できません。(153ページ 変数の長さの項参照)  
\$(ドル)のついた変数が文字変数です。  
30 LEN B\$(0)は文字の数を数えます。  
10010110なら文字の数は8です。

50 C\$=MID\$(B\$(0), N, 1)  
B\$(0)の文字列の  
左からN番目の文字から  
1文字取り出して  
C\$の中へ入れます。

60 VAL C\$では文字(C\$)を数字に変換して次の計算に備えます。

80 Nはパラメーターです。1ずつ加えて行けば、B\$(0)の文字列の左から1ずつ動かすことになります。L=0で最後の桁ですから計算を止めて、110ラインで答を表示します。

1111111=255. 11111111111111=65535 となることを確認してください。



GOTO 'A' ( ) GOTO 'C' ( )

例題②⑤、②⑥、②⑦をそのまま実行すれば、2進数  $\longleftrightarrow$  10進数  $\longleftrightarrow$  16進数 で基本的な変換

GOTO 'B' ( ) GOTO 'D' ( )

はできますが、ここで少し工夫してもらいたいことがあります。

1. 演算可能な桁数以上の数を入力した、などのミスを防ぐにはどうすればよいでしょうか。
2. 2進数  $\longleftrightarrow$  10進数変換で、ここでは16ビットまでの1の補数で表していますが、2の補数で表すにはどうすればよいでしょうか。
3. 表示を見やすいように工夫してください。答の表示では、10進数をSTR\$を使って'文字'に直したほうがよいでしょう。特に250、350でNの数値が変化していることに注意してください。

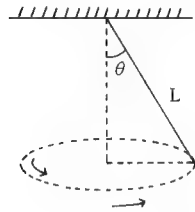
## 演習問題

8

9

を行ってください。

〔例題〕② 71ページの例題①をプログラム化した例題



①左図の円すい振り子の周期をT、糸の長さをL、糸の鉛直となす角を $\theta$ とすると、

$$T = 2\pi \sqrt{\frac{L \cdot \cos\theta}{g}}$$

の関係があります。今、糸の長さLを0.5[m]から5[m]まで、0.5[m]刻みで変化させたときのLとTの変化を表示するプログラムを作りなさい。ただし、 $g = 9.8[\text{m/sec}^2]$ 、 $\theta = 25^\circ$ とします。

②また、それぞれを表示した後で、Tの値が3秒を超えないLの最も大きい値を表示するプログラムを作りなさい。

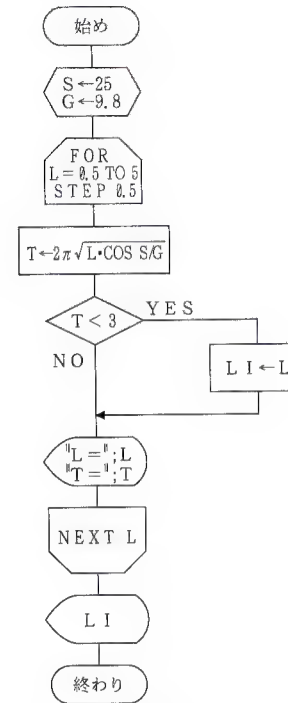
### ■プログラム例

```

10 REM フリコ
20 S=25:G=9.8
30 FOR L=0.5 TO 5 STEP 0.5
40 T=2*PI*SQR(L*COS S/G)
50 IF T<3 THEN LI=L
60 PRINT "L=";L;USING"###.##";T;"T=";T
65 FOR J=1 TO 1000:NEXT J
70 USING
80 NEXT L
90 PRINT "T<3 --- L=";LI;"(M)"
100 END

```

### ■フローチャート

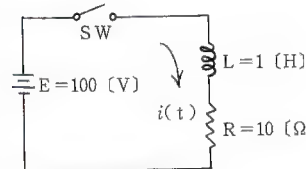


### ■結果

|       |        |
|-------|--------|
| L=0.5 | T=1.35 |
| L=1.  | T=1.91 |
| L=1.5 | T=2.34 |
| L=2.  | T=2.70 |
| L=2.5 | T=3.02 |
| L=3.  | T=3.30 |
| L=3.5 | T=3.57 |
| L=4.  | T=3.82 |
| L=4.5 | T=4.05 |
| L=5.  | T=4.27 |

T<3 --- L=2. (M)

〔例題〕② 72ページの例題③をプログラム化した例題



(a) 左図の回路において、スイッチSWを入れてから0.1秒ごとに1秒までの間に流れる電流を表示するプログラムを作りなさい。ただし、時間と電流を同一行に表示するようにしなさい。

$$i(t) = \frac{E}{R} (1 - e^{-\frac{R}{L}t})$$

(b) 次に、抵抗Rを10[Ω]から10[Ω]ずつ50[Ω]まで変化させたとき、回路を流れる電流が、 $i = \frac{E}{R}$  の62%から65%の範囲になる時間を求めるプログラムを作りなさい。ただし、時間は0秒から0.2秒まで0.002秒間隔で調べなさい。



## ■プログラム (a)

```

10 REM R=L
20 INPUT 'R='; R, 'L='; L, 'E='; E
30 FOR T=0 TO 1 STEP 0.1
40 I=E/R*(1-EXP(-R*T/L))
50 PRINT 'T='; T; 'I='; I; USING '##.###'; I
60 USING
70 NEXT T
80 END

```

## ■プログラム (b)

```

10 REM R=L
20 INPUT 'L='; L, 'E='; E
30 FOR R=10 TO 50 STEP 10
40 FOR T=0 TO 0.2 STEP 0.002
50 I=E/R*(1-EXP(-R*T/L))
60 IF (I>=E*0.62/R) AND (I<=E*0.65/R) THEN PRINT 'TR('; R; ')='; T; '(S)'; GOTO 80
70 NEXT T
80 NEXT R
90 END

```

## ■結果 (a)

```

T=0. I= 0.000
T=0.1 I= 6.321
T=0.2 I= 8.646
T=0.3 I= 9.502
T=0.4 I= 9.816
T=0.5 I= 9.932
T=0.6 I= 9.975
T=0.7 I= 9.990
T=0.8 I= 9.996
T=0.9 I= 9.998
T=1. I= 9.999

```

## ■結果 (b)

```

TR(10.)=0.098 (S)
TR(20.)=0.05 (S)
TR(30.)=0.034 (S)
TR(40.)=0.026 (S)
TR(50.)=0.02 (S)

```

### 3. 構造化BASIC命令の使いかた

構造化BASIC命令と呼ばれる次の4つの命令について説明します。これらの命令はプログラムをより見やすくわかりやすくするために考案されたものです。

#### (1) 判断 (IF~THEN~ELSE~ENDIF)

IF~THEN~ELSEは"もし~ならば~"や"もし~でなければ~"という意味で、いろいろな判断を行う命令です。

STEP 4で説明した"IF~THEN~ELSE"(1行形式)の他に"IF~THEN~ELSE~ENDIF"(ブロック形式)を用いて判断することができます。基本的な動作は1行形式のIF文と同じです。

ブロック形式のIF文は次の形で用います。

```

IF 条件式 THEN
  実行文
[ELSE
  実行文]
ENDIF

```

次の1行形式の命令文

```

10 IF A>=10 THEN PRINT A ELSE A=A+1

```

は、ブロック形式では次のように書き表されます。

```

10 IF A>=10 THEN
20   PRINT A
30 ELSE
40   A=A+1
50 ENDIF

```

IF文が成立すればIF文とELSEの間の命令を実行し、終了後ENDIFの次の行を実行します。成立しなければELSEが書かれている行の次の行から実行します。実行文が複雑になる場合は、ブロック形式にするとわかりやすくなります。

(注) ブロック形式か1行形式かの判断は、IFが行頭にあり、なおかつTHENの後に実行文が書かれているかどうかで行います。

#### (2) 繰り返し(REPEAT~UNTIL、WHILE~WEND)

STEP 5で説明した"FOR~NEXT"は繰り返し回数を最初に指定するのに対し、次の2つの命令はいつ終わるかわからない繰り返しに使用します。

"REPEAT~UNTIL"は条件が成立するまで繰り返し実行(条件を満足すると終了)するのに対し"WHILE~WEND"は条件が成立している間繰り返し実行します。

##### REPEAT~UNTIL

一度ループ内を実行し、最後に条件判断して繰り返します。

REPEAT~UNTIL文は次の形で用います。

```

REPEAT
  {
UNTIL 条件式

```

##### WHILE~WEND

ループに入る前に条件判断するため、一度もループ内を実行せずに終わる場合もあります。

WHILE~WEND文は次の形で用います。

```

WHILE 条件式
  {
WEND

```

### (3) 振り分け(SWITCH~CASE~DEFAULT~END SWITCH)

ON~GOTO命令は数値だけを扱うのに対し、SWITCH~CASE命令では、文字も扱うことができます。したがって、入力された変数をそのまま比較し振り分けるには、SWITCH~CASEの命令が便利です。たとえば、電子電話帳などのプログラム作成に利用すると便利です。

SWITCH~CASE~DEFAULT~ENDSWITCH文は次の形で用います。

SWITCH 変数

CASE { 式  
文字列 }

実行文

[CASE { 式  
文字列 }

実行文]

[DEFAULT

実行文]

ENDSWITCH

## 4. スクリーンエディタについて

BASICのPROモードやTEXTのエディット中ではスクリーンエディタ機能が働き、プログラムの修正や編集が容易になりました。ただし、BASICのRUNモードなどでは働きません。106ページで説明した円の面積を求めるプログラムを、スクリーンエディタ機能を使って球の体積を求めるプログラムに変更してみます。

LIST命令でプログラムリストを表示させます。

| キ ー 操 作             | 表 示 部                                                       |
|---------------------|-------------------------------------------------------------|
| PROモードにします。<br>LIST | 10: INPUT R<br>20: S= PI *R^2<br>30: PRINT S<br>40: END     |
| V  4  3  *<br>R  3  | 10 INPUT R<br>20 V=4/3*PI*R^3_<br>30 PRINT S<br>40 END      |
| R  V                | 10 INPUT R<br>20 V=4/3* PI *R^3<br>30 PRINT R, V_<br>40 END |

| キ ー 操 作  | 表 示 部                                                       |
|----------|-------------------------------------------------------------|
| 修正を終了します | 10 INPUT R<br>20 V=4/3* PI *R^3<br>30 PRINT R, V_<br>40 END |

スクリーンエディタ機能では1行ごとの修正だけでなく、プログラム全体を平面的に考えて修正することもできます。1行ごとにを押す必要はありません。ただし、修正の最後はを押してください。

を押さずに[CLS]を押すと、最後の行が修正されません。

なお、異なるプログラム行にカーソルを移動させたときや[BASIC]を押したときも、プログラム修正は行われます。

次にスクリーンエディタの中での主なキーの動作について説明します。

- ..... 1桁右にカーソルを移動  
行末にあるときは、次の行の先頭に移動
  - ..... 1桁左にカーソルを移動  
先頭にあるときは、前の行の行末に移動
  - ..... 桁位置を変えずに1行下にカーソルを移動  
行末よりも右側になるときは行末に移動
  - ..... 桁位置を変えずに1行上にカーソルを移動  
行末よりも右側になるときは行末に移動
- これらのキーは押し続けると連続して送っていきます。
- ..... プログラムの修正を終了します。

#### ①プログラム行の削除

プログラムの任意の行を削除するときは、[CLS]を押して表示を消した後、不要なラインナンバーの番号を入力してを押すか、またはラインナンバーだけを残して命令をすべて削除します。

〈例〉 不要なラインナンバー

なお、ラインナンバーも含めて削除([SHIFT] + [DEL]の操作)すると、削除したことにはなりません。連続した複数行のプログラムを削除するときは、DELETE命令を使用することもできます。

| キ ー 操 作 | 表 示 部                                                          |
|---------|----------------------------------------------------------------|
| LIST    | 10: INPUT R<br>20: V=4/3* PI *R^3<br>30: PRINT R, V<br>40: END |
|         | 10 INPUT R<br>20 V=4/3* PI *R^3<br>30 PRINT R, V<br>40 END     |

| キ ー 操 作                                    | 表 示 部                                                    |
|--------------------------------------------|----------------------------------------------------------|
| (SHIFT) + (DEL)<br>(10回押す)                 | —<br>20 V=4/3*PI*R^3<br>30 PRINT R, V<br>40 END          |
| ▼<br>元の10行目の内容があらわれます。<br>(削除したことにはなりません。) | 10 INPUT R<br>20 V=4/3*PI*R^3<br>30 PRINT R, V<br>40 END |

## ②プログラム行の複写 (追加)

プログラムリスト表示中に、ラインナンバーを書き換えるとプログラム行の複写ができます。元のプログラム行は残っており、複写したプログラム行の内容は変更することもできます。

複写 (ラインナンバーを書き換える) した後、異なるプログラム行にカーソルを移動させる (◀ または、▼、▲ の操作) と、プログラムは番号順に並び替わって表示されます。

新しいプログラム行を追加するときは、上のように複写してプログラム行を追加するか、または、(CLS) を押して表示を消した後、新しいプログラム行を追加します。なお、プログラムリストの最終行の次の行からはプログラム行の追加ができます。

連続した複数行のプログラムを複写するときは、LCOPY命令を使用することもできます。

# BASICによるプログラム演習問題

## 演習問題を始める前に

これからBASIC言語の演習問題を手がけていきましょう。演習問題を始める前に、次のことをよくもって解答してください。以後、演習問題は下の1、2、3項にもとづいて行うこととします。

1. 問題文をよく読んでプログラムの手順となる「流れ図」(フローチャート)を作る。
2. プログラムリストを別紙に書く (ただしプリンタをお持ちの方は除く)。
3. メモリ内容表を作成する。

(プログラムリスト参考例)

タイトルを必ず書く。

(メモリ内容表参考例)

| ○×△□のプログラム |             |
|------------|-------------|
| 10         | INPUT A     |
| ⋮          |             |
| 60         | T=(A+B+C)/2 |
| ⋮          |             |
| 150        | END         |

| メモリ内容表 |                       |
|--------|-----------------------|
| 変 数    | 内 容                   |
| A      | 三角形の一辺                |
| B      | 三角形の一辺                |
| ⋮      |                       |
| T      | $T = \frac{a+b+c}{2}$ |

●解答は先生にお聞きください。(指導マニュアルに解答例を記載しています。)

## 演 習 問 題 1

1. 半径Rを入力して円周ℓを求めるプログラムを作りなさい。

半径Rは2[cm]、4[cm]、6[cm]、8[cm]、10[cm]を入力するものとします。

|       |   |   |   |   |    |
|-------|---|---|---|---|----|
| R[cm] | 2 | 4 | 6 | 8 | 10 |
| ℓ[cm] |   |   |   |   |    |

2. 三角形の底辺Lと高さHを入力して面積Aを求めるプログラムを作りなさい。

L, Hの値は、それぞれ表の数値とします。

|                     |   |    |    |    |    |
|---------------------|---|----|----|----|----|
| L[cm]               | 5 | 10 | 15 | 20 | 25 |
| H[cm]               | 3 | 6  | 9  | 12 | 15 |
| A[cm <sup>2</sup> ] |   |    |    |    |    |

3. 水X[g]の中へ塩Y[g]を入れたときの濃度N[%]を求めるプログラムを作りなさい。

|                                |      |      |      |
|--------------------------------|------|------|------|
| X                              | 1000 | 1000 | 1000 |
| Y                              | 5    | 10   | 15   |
| $N = \frac{100 \times Y}{X+Y}$ |      |      |      |

4. 数X, Y, Zを入力してXY+YZ+ZXを求めるプログラムを作りなさい。

|    |     |    |     |    |    |
|----|-----|----|-----|----|----|
| X  | 1   | -2 | 3   | -4 | 5  |
| Y  | 6   | -7 | 8   | -9 | 10 |
| Z  | -11 | 12 | -13 | 14 | 15 |
| 結果 |     |    |     |    |    |

5. 数X, Y, Zを入力して $(XY^2 + YZ^2 + ZX^2)$ を求めるプログラムを作りなさい。

X, Y, Zの値は演習問題1の4.の値を用いなさい。

6. 数X, Yを入力して $(X/6 - Y/3)^2$ を求めるプログラムを作りなさい。

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| X  | 83 | 84 | 85 | 86 | 87 |
| Y  | 19 | 18 | 17 | 16 | 15 |
| 結果 |    |    |    |    |    |

## 演習問題 2

1. 123.619を入力して、(1)切り捨て(2)四捨五入をして、小数第2位まで求めるプログラムを作りなさい。
2. 数X, Yを入力して $X+Y$ を計算し、プラスなら1、マイナスなら-1、0なら0と表示するプログラムを作りなさい。

数値例

|    |   |    |   |    |    |
|----|---|----|---|----|----|
| X  | 1 | 3  | 5 | 4  | 0  |
| Y  | 2 | -5 | 6 | -4 | -3 |
| 結果 |   |    |   |    |    |

3. 数X, Y, Zを入力して $X*Y*Z$ を計算し、プラスなら1、マイナスなら-1、0なら0と表示するプログラムを作りなさい。

数値例

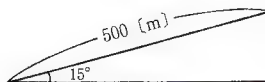
|    |   |    |    |   |    |
|----|---|----|----|---|----|
| X  | 1 | -1 | -1 | 0 | -5 |
| Y  | 2 | -2 | 2  | 2 | 2  |
| Z  | 3 | -3 | -3 | 5 | -3 |
| 結果 |   |    |    |   |    |

4. 2桁の乱数を2個作り、それらを表示し、その乱数の差の絶対値を求めるプログラムを作りなさい。
5. 2桁の乱数を2個作り、それらを表示し、その乱数の和、差、積、商を求めるプログラムを作りなさい。
6. X度を入力して $\sin(X) + \sqrt{3} \cdot \cos(X)$ の値を求めるプログラムを作りなさい。

数値例

|    |     |     |     |
|----|-----|-----|-----|
| X  | 15° | 30° | 45° |
| 結果 |     |     |     |

7.  $\sin(A+B) = \sin(A)\cos(B) + \cos(A)\sin(B)$ です。右辺と左辺を別々に計算して表示するプログラムを作りなさい。A=15°、B=30°とします。
8. 下図のように15度の坂道を歩くと何m高くなっているかを求めるプログラムを作りなさい。



## 演習問題 3

1. 数A, B, Cを入力して $A > B$ かつ $B > C$ なら $A*B*C$ の計算を、 $A > B$ かつ $B \leq C$ なら $A+B+C$ の値を、 $A \leq B$ なら $A/B+C$ の値を求めるプログラムを作りなさい。
2. 最初の日に1円、次の日に2円、次の日に4円と順に毎日倍額ずつ貯金をする100万円を超える日は何日目であるかを求めるプログラムを作りなさい。
3.  $1+2+3+\dots+X$ の合計が初めて200を超えるXの値を表示するプログラムを作りなさい。
4. 1, 3, 5, 7, 9, …, Xまでの和を求め、和が1000を超えないXの最大値を表示するプログラムを作りなさい。
5. X, Yを入力し、 $X=1$ で $Y=2$ なら'A'と、 $X=3$ で $Y=4$ なら'B'と、 $X=5$ で $Y=6$ なら'C'と表示し、上記以外の数字の組み合わせを入れたときはプログラムの最初に戻るプログラムを作りなさい。
6. 2つの数XとYを入力し、両方マイナスのときは $\sqrt{X*Y}$ を、どちらか一方がプラスのときは $X*Y$ を、両方プラスのときは $X/Y$ のそれぞれの値を表示するプログラムを作りなさい。
7. 数値を10個入力して合計と平均を求めるプログラムを作りなさい。

## 演習問題 4

1. 自然数1から100までの和を求めるプログラムを作りなさい。  

$$S = 1 + 2 + 3 + \dots + 100$$
2. 自然数MからNまでの和と平均を求めるプログラムを作りなさい。M, Nの値は入力するものとし $M < N$ とします。  

$$S = M + \dots + N$$
3.  $Y = 3X^3 + 2X^2 + X + 15$ において、Xの値を-10から10まで、0.5刻みで変化させてYの値を求めるプログラムを作り、結果を表に示なさい。
4. ① X(単位は度)を0度から360度まで10度刻みで変化させて $\sin X$ を求めるプログラムを作り0度から360度までの表を完成しなさい。  
 ②  $\cos X$ についても①と同じことをしなさい。  
 ③  $\tan X$ についても①と同じことをしなさい。ただし、 $\tan 90^\circ$ と $\tan 270^\circ$ のときは無限大となりエラーになりますので、'\*\*\*'のように表示させなさい。
5. 数Xを入力してXが正なら $\sin X$ を、Xが0なら $\cos X$ を、Xが負なら $\tan X$ を求めるプログラムを作りなさい。
6.  $AAB + BB = BAA$ つまり、 $(100 \times A + 10 \times A + B) + (10 \times B + B) = (100 \times B + 10 \times A + A)$ となるAとBを求めなさい。A, Bは1桁の整数とします。
7.  $ABA \times B = BCB$ つまり、 $(100 \times A + 10 \times B + A) \times B = (100 \times B + 10 \times C + B)$ となるA, B, Cを求めるプログラムを作りなさい。
8. 三角形の辺A, B, Cにおいて、それぞれを1から20まで変化させたとき、直角三角形となる組み合わせをすべて求めるプログラムを作りなさい。
9.  $Y = 6X^2 - 5X - 9$ の式において、 $-10 \leq X \leq 10$ の範囲でYの最大値を求めるプログラムを作りなさい。Xは0.2刻みとします。



10. データを10個入力して最大値と最小値を求めるプログラムを作りなさい。

5、7、-6、8、15、-4、-8、9、13、12

### 演習問題 5

- データ 2、4、6、8、10を読んで、その積を求めるプログラムを作りなさい。
- 次のデータを、200から順に引いて、答えを表示するプログラムを作りなさい。  
データ 24、8、29、35、10
- 次のデータのうち、最初の数それ以降のデータで割算をして、答えを表示するプログラムを作りなさい。  
データ 1982、6、12、25
- A=57、B=8、C=14となるように、データ57、8、14を読み込んでそれを表示し、 $A * C / B$ の答えを表示するプログラムを作りなさい。
- データ 5、6、7、8、9を読んで、それらをそれぞれA、B、C、D、Eに割りあてて、 $A * B$ 、 $(A * B) / (C + D)$ 、 $(A * E) - (B + C + D)$ の答えを表示するプログラムを作りなさい。
- 10人の身長を測定した結果、次のようになりました。10人の身長の測定データを読み込んで、平均身長を求めるプログラムを作りなさい。  
データ 165.5、170.5、172.3、168.4、182.6、  
159.9、174.8、167.6、177.8、173.6
- 次のデータを読んでA=35、B=29、C=8、D=8、E=49、F=19、G=8、H=49となるようにそれぞれ表示するプログラムを作りなさい。  
データ 35、29、8、49、19
- 10人に数学のテストを行ったところ、次のような結果でした。それぞれの点数を読んでから10人の平均点を求め、さらに平均点に最も近い点数を見つけ出し、平均点と平均点に最も近い点数を表示するプログラムを作りなさい。

| 番号    | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|-------|----|----|----|----|----|----|----|----|----|----|
| 点数(点) | 20 | 40 | 50 | 35 | 70 | 80 | 68 | 55 | 90 | 83 |

### 演習問題 6

- 数Xを入力して $X=1$ のとき $X^2$ 、 $X=2$ のとき $\sqrt{X}$ 、 $X=3$ のとき $X^3$ 、 $X=4$ のとき $X^4$ を表示するプログラムを作りなさい。
- 数Xを順次入力して、0が入力されたらそれまでの数の和と平方和を求めるプログラムを作りなさい。
- 数Xを入力して $(3X^2-3) + (3X^2-3)^2 - 6X$ を求めるプログラムを作りなさい。
- 数X、Y、Zを入力して $(6X^2+1) - (6Y^2+1) + (6Z^2+1)$ を求めるプログラムを作りなさい。

### 演習問題 7

- 〔例題〕⑩において、M個、N列のようなデータならプログラムをどのように作り替えたらいいか。そのプログラムを作りなさい。
- 〔例題〕⑪において、N人の生徒でK科目の場合の合計と平均を求めるプログラムに作り替えなさい。
- 〔例題〕⑫において、データ表の数値を入力した後、斜めのライン、1、12、23、34、45と5、14、23、32、41の合計を求めるプログラムを作りなさい。
- 下の表のような10人の生徒の得点を入力し、合計および平均、母標準偏差 $\sigma$ と偏差値Tiを求めるプログラムを作りなさい。

| 番号 | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   | 10 |
|----|----|----|----|----|----|----|----|----|-----|----|
| 得点 | 85 | 90 | 78 | 85 | 95 | 68 | 59 | 74 | 100 | 66 |

各人の得点を $X_1, X_2 \dots X_n$  (nは人数) とすると、

$$\text{合計 } \sum X_i = X_1 + X_2 + \dots + X_n \quad \text{平均 } \bar{X} = \frac{\sum X_i}{n} \quad (\text{ここではHと表現します})$$

$$\text{母標準偏差 } \sigma = \sqrt{\frac{\sum X_i^2 - nH^2}{n}} = \sqrt{\sum X_i^2 / n - H^2}$$

$$\text{偏差値 } Ti = 10 \times \frac{Xi - H}{\sigma} + 50$$

で求められます。

- 4の問題のデータを用いて低い点数から順に並べ替えをするプログラムを作りなさい。

### 演習問題 8

- 文字列"ABCDEFGHIJ"を入力してA、C、E、G、Iと1つ飛びにばらばらに表示するプログラムを作りなさい。
- 文字列"ABCDEFGHIJ"を入力してA、AB、ABC、ABCD、ABCDEと前の表示よりも1文字ずつ増えて表示するプログラムを作りなさい。
- 文字列"ABCDE54321"を入力して"12345EDCBA"と表示するプログラムを作りなさい。
- 文字列 I\$="I"、A\$="└AM┐"、A1\$="└A┐"、C\$(0)="└COMPUTER"を代入して、I AM A COMPUTERと表示するプログラムを作りなさい。
- 文字列 XS="I AM A COMPUTER"を入力して、その中の"COM"だけ表示するプログラムを作りなさい。

### 演習問題 9

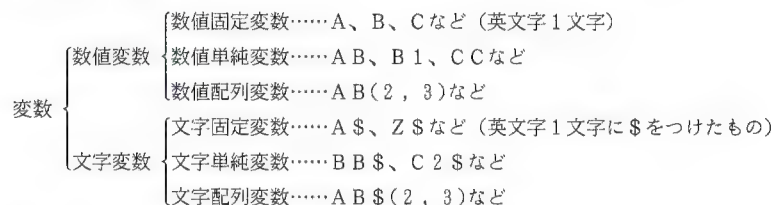
- 2つの文字列A\$="MAXABC"、B\$="XYMIN"を入力して文字数の大小を比較した後、大きい方の文字の頭文字3文字を表示するプログラムを作りなさい。
- 各自、自分自身の姓と名をそれぞれローマ字読みで入れ、その長さが何文字かを調べ、次に名を先に、姓を後に表示するプログラムを作りなさい。

## 5. 変数の種類と使いかた

これまで変数を使ってプログラムを作ってきましたが、ここで変数の種類と使いかたについて説明します。

### (1) 変数の種類

変数には大きく分けて、数値変数と文字変数の2種類があります。それらは、さらに固定変数、単純変数、配列変数に分類されます。



#### 【変数の名前】

変数の名前には、A～Zのアルファベット1文字か、あるいはAA、BC、A5などのように文字や数字を組み合わせた2文字が使用できます。

●変数名には次のような文字は使用できません。

- ①カタカナおよび記号（ただし、\$記号は文字変数を表す記号として使用されます。）
- ②予約語（予約語とは、本機がBASIC命令や関数命令などの名前として使用しているものです。）  
〈例〉PI、IF、TO、ON、SINなど
- ③小文字のアルファベット（小文字のアルファベットは大文字に変換されます。）

- 変数は必ずアルファベットで始まっていなければなりません。たとえばA5は変数名になりますが、5Aは変数名になりません。
- 変数名に3文字以上使用してもエラーにはなりませんが、本機が変数名と判断するのは最初の2文字だけです。たとえば、変数名にKOTAE1とKOTAE2を使用すると、これらを区別できずにKOという同じ変数と判断します。
- 固定変数（1文字で指定する変数）では、数値変数と文字変数に同じ名前を使用できません。たとえば、数値変数としてAを使用しているとき、文字変数A\$を使用することはできません。（この場合、後からA\$に文字を代入すると、前の数値変数Aが消されます。）

#### 【変数の長さ】

変数はその種類によって、長さ（その変数に格納できるデータの最大の長さ）が決められています。次に各変数の長さを示します。

| 変数の種類  | 変数の長さ                   |
|--------|-------------------------|
| 数値変数   | 有効数字10桁まで（仮数部10桁、指数部2桁） |
| 文字固定変数 | 7文字まで                   |
| 文字単純変数 | 16文字まで                  |
| 文字配列変数 | 16文字（ただし、1～255文字まで設定可能） |

- それぞれの文字変数の長さを超える文字を記憶させようとした場合、超えた分の文字は無視（切り捨て）されます。

### (2) 固定変数

これまでの説明やプログラム例で、変数名をA、B、C…またはA\$、B\$、C\$…と指定して使用しましたが、これらの変数を固定変数（1文字変数）と呼びます。

固定変数は、変数として使用されるエリア（区域）がメモリ上に独立して確保されており、このエリアにはプログラムなどが書き込まれることはありません。

このエリアをデータ専用エリアと呼びます。

また、同じ名前の数値固定変数と文字固定変数（たとえばAとA\$）は同じ場所が使用されます。

### (3) 単純変数

単純変数は変数名をAAやB1のように2文字（あるいはそれ以上）で指定する変数です。

この変数は固定変数のように使用される場所は決まっておらず、初めて使用したときにメモリ内（プログラム・データエリア内）に自動的に確保されます。

また、同じ変数名でも数値単純変数と文字単純変数は別々に確保されるので、たとえばABとAB\$を同時に使用することもできます。

### (4) 配列変数（一次元配列、二次元配列）

数学などで同じ性質の複数個のデータを表すとき、たとえば、

$$X_1, X_2, X_3, \dots, X_n$$

のように、1つの変数名に添字をつけて表す場合があります。

BASICでも同様に、1つの変数名に（ ）で囲った添字をつけて、次のように表すことができます。

$$X(1), X(2), X(3), \dots, X(10)$$

このように添字をつけて表す変数を配列変数と呼びます。

配列変数を使うとデータの集計などの作業がしやすくなります。

配列変数を使用するときは、事前にDIM（ディメンジョン）命令によって、配列名とその大きさを定義（宣言）しておかなければなりません。（くわしくは、BASICの各命令の説明のDIM命令参照）

DIM命令で定義することによって、その大きさの配列変数がメモリ（プログラム・データエリア）上に確保されます。

#### 【一次元配列】

添字が1個だけのものを一次元配列といいます。たとえば、

$$\text{DIM } X(5)$$

を実行すると、次のように配列名Xの6個の変数（配列要素）が確保されます。

|      |      |      |      |      |      |
|------|------|------|------|------|------|
| X(0) | X(1) | X(2) | X(3) | X(4) | X(5) |
|      |      |      |      |      |      |

（注）この図は配列が横に並んでいるように書いていますが、縦に並んでいると考えても同じです。実際の使いかたは129ページを参照してください。

## 【二次元配列】

本機は添字を2個まで使用できます。添字が2個のものを二次元配列といいます。一次元配列が、いうなれば横一列（または縦一列）の配列だったのに対し、二次元配列は縦と横の配列といえます。下の図は、

```
DIM X(3, 5)
```

と定義したときに確保される変数（配列要素）を表したものです。24個の変数が確保されます。

|        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|
| X(0,0) | X(0,1) | X(0,2) | X(0,3) | X(0,4) | X(0,5) |
|        |        |        |        |        |        |
| X(1,0) | X(1,1) | X(1,2) | X(1,3) | X(1,4) | X(1,5) |
|        |        |        |        |        |        |
| X(2,0) | X(2,1) | X(2,2) | X(2,3) | X(2,4) | X(2,5) |
|        |        |        |        |        |        |
| X(3,0) | X(3,1) | X(3,2) | X(3,3) | X(3,4) | X(3,5) |
|        |        |        |        |        |        |

## 【文字配列変数】

配列変数にも文字変数があり、配列名に\$（ドル）記号をつけて表します。

```
DIM Z$(9)          DIM X1$(2, 1)
DIM Y$(5, 4)        X1$(0, 1) = 'ヨコハマ'
```

## ●文字配列変数の拡張

文字固定変数は1個の変数に最大7文字、文字単純変数は1個の変数に最大16文字記憶できる固定長の変数でしたが、文字配列変数は1～255文字の範囲で、任意に変数の長さを指定できます。

```
DIM C$(9)*30
C$(0)~C$(9)の各変数には、それぞれ最大30文字まで記憶できます。
DIM N$(5, 4)*6
N$(0, 0)~N$(5, 4)の各変数には、それぞれ最大6文字まで記憶できます。
```

このように、DIM文に\*式をつけて、変数の長さを指定します。

なお、長さを指定しない（\*式がない）ときは自動的に16文字が指定されます。

## 【メモリの構成と変数】

プログラム、単純変数、配列変数が記憶される領域をプログラム・データエリアと呼びます。プログラムはプログラム・データエリアの前方から、データは後方からエリアを確保します。そのためにプログラムの長さによって変数として使える大きさが変わってきますので注意が必要です。なお、固定変数はデータ専用エリアとして別に確保されています。

次に、各変数を確保する場合に使用するバイト数およびプログラムの各命令などの占めるバイト数を記しておきますので参考にしてください。



## 変数の場合

| 変 数    | 変数の名前 | デ ー タ     |
|--------|-------|-----------|
| 数値単純変数 | 7バイト  | 8バイト      |
| 数値配列変数 |       |           |
| 文字単純変数 | 7バイト  | 16バイト     |
| 文字配列変数 | 7バイト  | 指定されたバイト数 |

- たとえば、DIM B\$(2, 3)\*10と指定した場合は、次のバイト数を使用します。  
7バイト（変数名）+10バイト（文字数）×{(2+1)×(3+1)}個=127バイトになります。

## 命令の場合

| 構成要素   | ラインナンバー | 命令文、関数 |  、その他 |
|--------|---------|--------|------------------------------------------------------------------------------------------|
| 使用バイト数 | 3バイト    | 2バイト   | 1バイト                                                                                     |

- たとえば、下のプログラム行を入力した場合、次のバイト数を使用します。  
10 PRINT A; 'X—'  
3バイト（行番号）+2バイト（命令文）+1バイト（）+6バイト（その他1×6）=12バイト
- プログラム・データエリアの残りのバイト数（フリーエリア）はFRE命令で求めることができます。（くわしくは、BASICの各命令の説明のFRE命令を参照してください。）

## 【変数をクリア（消去）するには】

- ① 個々の変数の内容を消去するには、次のようにします。

<例>    A = 0                      } 数値変数は0を代入することにより消去します。  
          B(1) = 0                }  
          A\$ = ''                 } 文字変数は''を代入することにより消去します。  
          B1\$ = ''                } ''を代入することは文字変数を文字が入っていない状態にするということです。この状態を Null（ヌル）と呼びます。

- 変数の内容を消去してもフリーエリアの大きさは変わりません。次のCLEAR命令やERASE命令を実行するとフリーエリアはふえます。

## ② CLEAR命令

すべての変数を一度に消去するには、CLEAR命令を実行します。

CLEAR命令の一般形は次のようになります。

```
CLEAR
```

この命令を実行すると、メモリ内に確保されていた配列変数や単純変数は消去され、固定変数の内容も消去されます。

- この命令ではプログラムは消去されません。配列変数や単純変数を消去することにより、フリーエリアがふえます。

## ③ ERASE命令

配列変数のみを消去します。

ERASE命令の一般形は次のようになります。

```
ERASE 配列名 [, 配列名, ……]
```

この命令を実行すると、指定された配列名の配列変数を消去します。なお、配列名の指定では、配列の大きさ（BASICの各命令の説明のDIM命令参照）を指定する必要はありません。






## 6. デバッグ

プログラムを実行したときに、何かの誤りで迷走したり思わぬ結果が出たりすると、プログラムリストを調べて誤りを探しますが、それだけではなかなか見つけにくいことがあります。

このようなときは、プログラムを1行ずつ実行させ、その経過をたどりながら誤りを探していくと見つかりやすくなります。

ここでは、1行ずつ経過をたどりながらプログラムを実行する方法について説明します。(この方法をトレースといいます。また、プログラムの誤りを探し、修正することをデバッグ(虫取り)といいます。)









## (1) デバッグのしかた

- ① RUNモードにしてください。トレースによるデバッグはRUNモードで行います。
- ② TRON  と押して、トレースモードにします。
- ③ RUN  と押してプログラムの実行を開始します。(最初の行の実行が終われば、実行した行番号を表示して実行が停止します。)
- ④ その後は  を押します。(1行だけ実行して停止します。) INPUT命令でのデータ入力、通常のプログラム実行と同じように  を押します。
- ⑤ プログラムの実行順序の確認や、各行(各ライン)実行後の変数の内容確認などを行いながらトレースを進め、プログラムが正しく実行されているかどうかチェックします。  
正しく実行されないときは、その原因を探して修正します。
- ⑥ デバッグが終了すれば、TROFF  と押してトレースモードを解除します。




次に簡単な例を示します。

```
<例> 1 0 INPUT 'A='; A, 'B='; B
      2 0 C=A*2
      3 0 D=B*3
      4 0 PRINT 'C='; C; 'D='; D
      5 0 END
```

[キー操作] RUNモード

|                                                                                               |                         |                |             |
|-----------------------------------------------------------------------------------------------|-------------------------|----------------|-------------|
| TRON         |                         |                |             |
| RUN         | A=                      |                | } INPUT命令実行 |
| 8  (データ入力) | B=                      |                |             |
| 9  (データ入力) | 1 0 :                   | ← 10行終了        |             |
|            | 2 0 :                   | ← 20行終了        |             |
|            | 3 0 :                   | ← 30行終了        |             |
|            | C=16, D=27, ← PRINT命令実行 |                |             |
|                                                                                               | 4 0 :                   | ← 40行終了        |             |
|            | >                       | 実行終了 (プロンプト表示) |             |

トレース中で行番号を表示しているときは、マニュアル操作で変数の内容呼び出し、予定した値になっているかどうかチェックできます。






このとき  を押せば、押している間、止まっている行の内容を表示します。(  を押して離れたときはプロンプト記号(>)が表示されますが、 で続けて実行できます。)

 を押したままにすると、連続的にトレースが実行されます。

- トレース中に、LOCATE命令で指定された画面の位置に結果などが表示された場合、次の行番号は、結果などが表示された行の次の行から表示されます。(LOCATEについては、BASICの各命令

の説明を参照)

- LOCATE命令で表示開始位置が指定されているときに、マニュアル操作で変数の呼び出しや計算などを行うと、表示開始位置の指定は解除されます。


(注) トレースモードは、TROFF  と押すか、 +  (または   ) と押す、または電源を切るか、オートパワーオフ(自動節電機能)で電源が切れるまで設定状態が保持されます。

## (2) プログラムの途中で実行を停止させてチェックする場合

トレースモードを設定しない場合でも、プログラムの実行を停止させたい位置にSTOP命令を書いておけば、STOP命令を実行した時点でブレークメッセージ(BREAK IN 行番号)を表示して、プログラムの実行が停止します。

このとき、


- ① マニュアル操作で変数の内容をチェックする。

- ② 続いて  の操作で、以降の行を1行ずつトレースする。

などの操作でデバッグを行います。その後、通常の実行状態に戻すときは

CONT 

と押します。

- 通常のプログラム実行中に  を押すと、現在実行している行の終わりで実行を停止し、ブレークメッセージを表示します。このときも前記と同様の操作を行うことができます。

なお、ブレーク状態(停止状態)のときに  を押せば、押している間停止しているプログラム行が表示されます。

# 7. プログラムのファイル

本機は、プログラムをファイルとして内部メモリに保存しておくことができます。

メモリの一部をプログラム・データエリアと切り離して、プログラムファイルエリアとして確保し、このエリアにプログラムを保存します。

本機で実行できるプログラムは、プログラム・データエリアに入っているプログラムです。プログラムファイルエリアのプログラムを実行したいときは、プログラムファイルエリアからプログラム・データエリアに呼び出してから実行します。

ここでは、BASICプログラムのファイルのしかたについて説明します。

(注) ● BASICプログラムだけではなく、テキストのファイルもあります。TEXTモードの説明も参照してください。

- プログラムを記憶するプログラムファイルエリアと、データを記憶するラムデータファイルエリア(シーケンシャルデータ)があります。これらはそれぞれ独立して確保されます。なお、ラムデータファイルエリアについては、次項の「8. データのファイル」を参照してください。



## (1) ファイルに関する命令

ファイルに関する命令を簡単に説明します。くわしくは、BASICの各命令の説明を参照してください。

| 〈命令〉   | 〈機能〉                                             |
|--------|--------------------------------------------------|
| FILES  | 登録されているファイルのファイル名を表示します。                         |
| LFILES | 登録されているファイルのファイル名をプリンタで印字します。                    |
| KILL   | ファイルを消去します。                                      |
| LOAD   | BASICプログラムをプログラムファイルエリアからプログラム・データエリアに呼び出します。    |
| SAVE   | プログラム・データエリア内のBASICプログラムをプログラムファイルエリアに保存（登録）します。 |

### ●ファイル名

KILL、LOAD、SAVE命令を実行するときはファイル名を指定する必要があります。ファイル名はファイルの見出しのようなものです。

ファイル名は最大8文字で構成され、次の文字が使用できます。

A～Z、a～z、0～9、#、\$、%、&、'、(、)、-、@、{、}、\_、°、カタカナ

また、ファイル名には拡張子をつけることができます。拡張子はファイルの種類を区別するためなどに利用します。拡張子はピリオドと3文字以内の文字で構成され、ファイル名の直後につけます。使用できる文字は、ファイル名と同じものです。（スペースは使用できません。）

なお、拡張子を省略すると自動的に「.BAS」が指定されます。

### ●ファイル名の完全な記述は次の形式になります。

'ファイル名.拡張子'

## (2) プログラムの登録（保存）（SAVE命令）

プログラム作成後、そのプログラムを登録する場合は、RUNモードまたはPROモードで次の操作を行います。

〈例〉 SAVE 'TEST' 

↑  
ファイル名

ファイルとして登録する場合、必ずファイル名が必要です。ファイル名には、拡張子をつけることができます。

この例では拡張子を省略していますが、省略すると“.BAS”が自動的につけられます。



なお、プログラムファイルエリアはSAVE命令でプログラムを登録したとき、自動的に必要な大きさが確保されます。ただし、メモリの残りが少なく、必要な大きさが確保できないときはエラー60になります。

## (3) ファイルの登録の確認（FILES、LFILES命令）

プログラムを登録した場合、FILESまたはLFILES命令で確認ができます。

〈例〉 FILES 

と操作すれば、登録されているファイル名が画面に表示されます。

登録されているファイルが多い場合は  を押していくことにより、画面に呼び出すことができます。戻すときは  を押します。

〈例〉 LFILES 

と操作すれば、登録されているファイル名が印字されます。（ただし、プリンタCE-126Pが接続されているときのみ有効です。）

## (4) プログラムの呼び出し（LOAD命令）

登録したプログラムは次の操作で呼び出すことができます。

〈例〉 LOAD 'TEST' 

この場合、ファイル名が“TEST. BAS”のファイル内容（プログラム）が呼び出されます。

登録したときに拡張子をつけたときは、拡張子まで完全に入力してください。ただし、拡張子が“.BAS”の場合は拡張子を省略できます。

FILES命令で呼び出した後、ファイル名を選んで呼び出すこともできます。

〈例〉 FILES  ファイル名を表示させます。

 ... 

 ... 

(SHIFT) + 

} 画面のファイル名の前にある矢印 (→) を、呼び出したいファイル名に移します。

矢印 (→) で示したファイルの内容（プログラム）が、プログラム・データエリアに呼び出されます。

### ご注意

拡張子をつけて登録する場合、拡張子には“.TXT”を使用しないでください。“.TXT”を使用すると、テキストプログラムのファイルと区別がつかなくなります。（177ページ参照）

## (5) 記録されているファイルの消去（KILL命令）

ファイルを消去するときは、次の操作を行います。

〈例〉 KILL 'TEST. BAS' 

この場合、“TEST. BAS”というファイルが消去されます。

KILL命令では、記録されているファイル名のとおり、ファイル名を拡張子まで入力してください。

ただし、拡張子が“.BAS”の場合は拡張子を省略できます。

なお、TEXTモードで消去する方法もあります。（第5章を参照）

# 8. データのファイル

本機は、データをシーケンシャルファイルとして内部メモリに保存しておくことができます。この領域をラムデータファイルエリアと呼びます。

ラムデータファイルエリアは、あらかじめINIT命令でファイルの大きさを確保しておかなければなりません。ラムデータファイルの使いかたについては、「第5章 2. 9 ラムデータファイル」を参照してください。

ラムデータファイルのデータはプログラム・データエリア内のプログラム（BASICプログラム）で、入出力（書き込みや読み出し）ができます。また、テキストエリアとの間でファイルの相互転送もできます。

## 9. 別のポケコンへの記録、読み込み

EA-129C（ポケコン接続ケーブル）ともう1台のPC-G850VSをお持ちの場合は、プログラムを別のポケコンに記録したり、読み込んだりできます。ここでは、プログラムを別のポケコンに記録する、あるいは別のポケコンから読み込む場合の操作方法について説明します。

### (1) ポケコン間通信に関する命令

次にポケコン間通信に関する命令を簡単に述べます。くわしくは個々の命令の説明を参照してください。

| 〈命令〉    | 〈機能〉                                              |
|---------|---------------------------------------------------|
| BLOAD   | 別のポケコンに記録されているBASICプログラムを読み込みます。                  |
| BLOAD ? | 別のポケコンに記録されているBASICプログラムと計算機内のBASICプログラムの照合を行います。 |
| BLOAD M | 別のポケコンに記録されている機械語プログラムを読み込みます。                    |
| BSAVE   | 計算機内のBASICプログラムを別のポケコンに記録します。                     |
| BSAVE M | 計算機内の機械語プログラムを別のポケコンに記録します。                       |

### (2) 準備


本機、EA-129C、もう1台のポケコンを、それぞれを接続してください。接続するときは両方のポケコンとも電源を切ってから行ってください。

なお、EA-129Cは両方のポケコンの左の周辺機器接続端子（11ピン）に接続します。


### (3) 別のポケコンへの記録方法

①本機にBASICプログラムを入れます。

②別のポケコンで読み込み命令を入力し実行します。

〈例〉 RUNモード（またはPROモード）を指定  
BLOAD 

③本機で記録命令を入力し実行します。

〈例〉 RUNモード（またはPROモード）を指定  
BSAVE 


●記録が始まると別のポケコン画面の下の行の右端桁に\*マークが表示されます。

④記録が終了すると、プロンプト記号が表示されます。続いて次項の方法で“照合”を行ってください。


### (4) 本機内と別のポケコンのBASICプログラムの照合

BASICプログラムを別のポケコンに記録した後、まちがいがなく記録されたかどうか照合して確認します。

①本機で照合命令を入力し実行します。

〈例〉 RUNモード（またはPROモード）を指定  
BLOAD? 

②別のポケコンで記録命令を入力し実行します。

〈例〉 RUNモード（またはPROモード）を指定  
BSAVE 

●BASICプログラムが見つかり、照合が始まると本機画面の下の方の右端桁に\*マークが表示されます。

③両方の内容がすべて一致していれば実行を終了し、プロンプト記号が表示されます。

もし、エラー82になった場合は、もう一度最初から照合を行ってください。

それでもなお、エラーになる場合は、もう一度“記録”から行ってください。

### (5) 別のポケコンからの読み込み

BASICプログラムを別のポケコンから本機に読み込む場合は、照合命令を読み込み命令にかえて、照合の場合と同じ手順で行ってください。

読み込み命令：BLOAD

〈例〉 BLOAD 

もし、読み込み途中でエラー80になった場合は、もう一度最初から読み込みを行ってください。


## 10. プログラムの実行開始方法とラベルについて

### (1) プログラムの実行開始方法

プログラムの実行開始方法には次の方法があります。

#### RUN命令によるもの

RUN  .....プログラムの先頭から実行開始

RUN 行番号  .....指定した行番号から実行開始

RUN ラベル  .....指定したラベルが書かれている行から実行開始

## GOTO命令によるもの

GOTO 行番号 [ ] …指定した行番号から実行開始

GOTO ラベル [ ] …指定したラベルが書かれている行から実行開始

ラベルは、'|' (ダブルクォーテーション) で前後を囲うか、\* (アスタリスク) を前につけて指定します。(くわしくは、次の「ラベルについて」の項参照)

```
<例> RUN "AB"
      GOTO *AB
```

これらの開始方法により、状態の解除や変数の消去などに違いがあります。

| RUN命令による実行                                                                                                                                                                                                                                                                                                                                                     | GOTO命令による実行                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>・ウェイト (WAIT) 指定 → 0 に設定</li> <li>・表示フォーマット (USING) 指定 → 解除</li> <li>・配列変数 (DIM 指定)、単純変数 → 消去</li> <li>・READ 文に対する DATA → 初期化する</li> <li>・LINE、GCURSOR の位置指定 → 初期化する</li> <li>・PRINT=LPRINT の指定 → 解除</li> <li>・すべてのファイル → 閉じる</li> <li>・パラレルポート → 閉じる</li> <li>・固定変数の内容 → 保持</li> <li>・トレースモード (TRON) 設定 → 保持</li> </ul> | <ul style="list-style-type: none"> <li>・ウェイト (WAIT) 指定 → 保持</li> <li>・表示フォーマット (USING) 指定 → 保持</li> <li>・配列変数 (DIM 指定)、単純変数 → 保持</li> <li>・READ 文に対する DATA → 初期化しない</li> <li>・LINE、GCURSOR の位置指定 → 初期化しない</li> <li>・PRINT=LPRINT の指定 → 保持</li> <li>・すべてのファイル → 閉じない</li> <li>・パラレルポート → 閉じない</li> <li>・固定変数の内容 → 保持</li> <li>・トレースモード (TRON) 設定 → 保持</li> </ul> |

(注) プログラムをRUN命令で実行したとき、配列変数や単純変数は消去されます。データを残しておきたいときは、GOTO命令で実行を開始してください。

## (2) ラベルについて

次の例のように、ラベルはプログラムの行の先頭 (行番号の次) に見出しとして書いておくものです。GOTOやGOSUB、THENなどのジャンプ先としてラベルを指定すれば、実行時に指定したラベルを探して、そこへジャンプします。

```
<例> 10  'AB': INPUT  ...
      :           ↑ ラベルAB
      50  *CDE: CLEAR  ...
      :           ↑ ラベルCDE
      100  IF  ... THEN 'AB'  ←ラベルABへジャンプ
      :
      150  GOTO  *CDE  ←ラベルCDEへジャンプ
```

また、プログラム実行開始命令であるRUNやGOTO命令で指定すれば、指定したラベルのある行からプログラムを実行できます。たとえば、使用できる行番号の範囲内で複数のプログラムの先頭にラベルをつけて本機に書き込んでおき、ラベルを指定して実行を開始すれば、必要なプログラムを実行できます。

## '|'で囲んだラベルおよびラベル指定で利用できる文字

'|'で囲んだラベルには、英字、数字、カタカナ、記号などが使用できます。

```
<例> 'ABCDE'
      'X10'
      'サブ ルーチン'
```

## \*を用いるラベルおよびラベル指定で利用できる文字

\*記号を先頭につけるラベルには、英字か、英字とそれに続く数字が使用できます。

```
<例> *START
      *S123
```

- ラベルは必ず英文字で始まっていなければなりません。
- 英字の小文字は大文字に変換されます。
- 予約語のスペルで始まる文字列は使用できません。
- カタカナや記号は使用できません。

本機では'|'で囲んだラベルと、\*を用いるラベルを混合して使うことができますが、1つのプログラムの中では、どちらかに統一されることをお勧めします。

なお、\*を用いるラベルは多くのパソコンでも用いられています。

## n進演算機能

n進演算機能では、10進、2進、16進について、それぞれの変換 (基数変換)、補数変換、および計算ができます。(符号付き15ビットの演算)

n進演算機能は、ROM内に格納しているn進演算のBASICプログラムをプログラム・データエリアに呼び出して使用します。

呼び出したプログラムはPROモードで変更することもできます。

## 【プログラムのスタート】

RUNモードで[SHIFT] + [BASE-n] と押してください。

プログラムが呼び出され、同時にプログラムがスタートして、右の画面 (初期画面) になります。

```
***** n シン エンサ*ン *****
1: ニュウリョク      2: ヘンカン
3: ホスウ            4: ケイサン

(1, 2, 3, 4)?
```

[1] ~ [4] を押して、それぞれの機能を選びます。

[SHIFT] + [BASE-n] と押したときに、BASICプログラムエリアにプログラムがある場合は、右のように表示されます。

```
BASIC DELETE OK? (Y)
```

n進演算を実行したいときは、BASICプログラムを消去しなければなりません。

BASICプログラムを消去するときは[Y]を押します。164ページの初期画面になります。

消したくないときは[CLS]など[Y]以外のキーを押してください。RUNモードになります。

(注) ● n進演算プログラムは、2794バイトの大きさがありますので、空きエリアが2794バイト以上ないと実行できません。また、実行時に変数として167バイト使用します。

プログラムを消去しても空きエリア(2794バイト必要)が不足するようなときは、前記操作で[Y]を押しても、プログラムの消去を行わずにRUNモードになります。

プログラムは存在しないが空きエリアが足りない(データが多い)ときも、[SHIFT] + [BASE-n] と押しても何も実行せずにRUNモードになります。

なお、実行時に変数が確保できない(167バイト必要)とエラー60になります。

このような場合は、変数を消去したり、TEXTプログラムを消去したりして、2961(2794 + 167)バイト以上空けてください。

- [SHIFT] + [BASE-n] と押すと、WAIT指定、USING指定、トレースモード(TRON)は解除されます。



#### ROMについて

ROMとは Read Only Memory のことで、読み出し専用のメモリです。関数やBASICなどのソフトウェアが格納されているところです。

プログラムの実行を中止するときは、[BREAK]を押します。再度、最初からスタートするときはRUN命令で実行を開始してください。[SHIFT] + [BASE-n] と操作すると、呼び出されているn進演算プログラムを消去して、再度呼び出すことになります。

なお、n進演算のプログラムは、約5分間キー操作をしないと自動的に実行を中止して、プロンプト記号(>)を表示します。

#### 【プログラムの使いかた】

##### 1. 概要

初期画面では、[1] ~ [4] で、入力、変換、補数、計算の4つの機能を選ぶことができます。

- 1: 入力……変換や計算の元になる数値を入れます。
- 2: 変換……表示している数値を次の順番で変換します。また、何進数(10進、16進、2進のいずれか)で入力や計算を行うかも指定します。  
10進数 → 16進数 → 2進数 → 10進数 …
- 3: 補数……表示している数値の結果の補数を求めます。
- 4: 計算……10進、16進、2進の四則計算(+ - \* /)と、論理計算(AND、OR、NOT、XOR)ができます。

このプログラムで扱える数値はそれぞれ、次のようになります。

2進……16桁以内の2進数値(16ビット目は符号)

17桁以上入力したときは、先頭から16桁が有効になります。また、計算結果が16桁を超える場合は、超えた分、上位桁が切り捨てられます。

16進……0 ~ F F F Fの16進数値(8 0 0 0 ~ F F F Fは負数)

5桁以上の16進数値を入力した場合は、後ろ4桁が有効になります。計算結果が4桁を超える場合は、超えた分、上位桁が切り捨てられます。

10進……-32768 ~ 32767の10進数値

10進数は、内部で16進数に変換されて処理されます。したがって、16進数に変換されたときに、上記の16進数の範囲になる値を扱います。

[補足]

- ・変換結果などを表示しているときに[1]を押すと、表示している進数での入力画面になります。
- ・入力した内容が、そのときに処理または計算できない値や数字、文字、桁数のときは、ERRORを表示した後、入力待ちの状態に戻ります。
- ・16進数のA ~ Fは英文字のキーで入力します。

##### 2. 実行例

プログラムを実行させて、初期画面にしてください。

〈例1〉

10進数の1230を16進数および2進数に変換します。

“入力”を選びます。

[1]

10進数の入力待ちになります。

(初期画面のときは10進数のモードになっています。)

1230を入力します。

1 2 3 0 [↵]

“変換”を選びます。

[2]

16進数に変換されます。

もう一度“変換”を選びます。

[2]

2進数に変換されます。

もう一度“変換”を選ぶと10進数に戻ります。

〈例2〉

16進数12C7の補数を求めます。

“変換”により、16進を選びます。

[2]

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[1 0 シン] = _
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[1 0 シン] = 1 2 3 0
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[1 6 シン] = 0 4 C E
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[ 2 シン] = 0 0 0 0 0 1 0 0 1 1 0 0 1 1 1 0
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[1 6 シン] = 0 4 C E
```



12C7を入力します。

[1] (入力を選びます)

12C7 [4]

[1]を押して“入力”を選ぶことを忘れないようにしてください。

補数を求めます。

[3]

もう一度、[3]を押して補数変換をすれば、ED39が12C7に戻ります。(ED39と12C7は互いに補数の関係です。)

## 参考

ここでいう補数は、2進数においては2の補数、16進数においては16の補数です(基数に対する補数)。10進数においては、符号を反転させて補数としています。

この場合の補数の関係は、互いを加えると0になる関係です(ただし、最上位桁に生じる桁上げは無視するものとする)。

たとえば、12C7HとED39Hを加えると10000Hになります。このとき、最上位桁の1を無視すると0000Hになります。(Hは16進数を示す記号)

〈例3〉

16進数3E7Cと0FF0のAND(論理積)を求めます。

16進を選び、3E7Cを入力します。

[2] (16進にします)

[1] 3E7C [4]

“計算”を選びます。

[4]

計算命令(AND)を選び、0FF0を入力します。

[A]

0FF0

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[16シン] = 12C7
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[16シン] = ED39
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[16シン] = 3E7C
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[16シン] 3E7C
(+, -, *, /, A, O, N, X) ?
```

+, -, \*, /……加算、減算、乗算、除算

A……AND O……OR

N……NOT X……XOR

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[16シン] 3E7C
AND 0FF0
```

実行します。

[4]

0E70が求められます。

〈例4〉

16進数1E01から10進数の96を引き、結果を16進数で求めます。

1E01を入力して、10進数に変換します。

[2] (16進にします)

[1] 1E01 [4]

[2] [2]

計算をします。

[=] 96 [4]

“計算”を選ぶ[4]の操作は省略しています。

結果を16進数に変換します。

[2]

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[16シン] = 0E70
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[10シン] = 7681
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[10シン] = 7585
```

```
***** n シン エンサ ン *****
1: ニュウリョク 2: ヘンカン
3: ホスウ 4: ケイサン

[16シン] = 1DA1
```

## 参考

次のファイル名でプログラムをプログラムファイルエリアに登録しておけば、[SHIFT]+[BASE-n]の操作で、呼び出し、実行ができます。

“BASE\_N.BAS” (注) \_はアンダーラインです。

プログラムファイルエリアにこのファイル名のプログラムがあると、内部のn進演算プログラムは呼び出されません。

したがって、n進演算プログラムを変更して使用するときは、このファイル名で変更したプログラムを登録しておけば、元のプログラムと同じように[SHIFT]+[BASE-n]と押して呼び出し、実行することができます。(ただし、元のプログラムを呼び出すときは、このファイルを消去する必要があります。)

なお、[SHIFT]+[BASE-n]と押した場合、自動的に GOTO 1000 を実行してプログラムをスタートさせます。したがって、プログラムは100行から書いてください。

# 第5章

## TEXTモード (テキストエディタ)

TEXTモード (テキストエディタ) では、アスキー形式でのプログラムの入力、編集、SIOへの入力やラムデータファイル (データファイル) の確保などが行えます。

本機のBASICの各命令は、中間コードと呼ばれる2バイトコードに変換して記憶しています。このコードはハードウェアやBASICインタプリタにより異なるため、そのままではパソコン等との通信はできません。

アスキーコードは、アルファベットや数字、基本的な記号がほとんどの機種で共通であるため、パソコン等ではアスキーコードでの通信が多く行われています。

本機では、パソコン等との通信を容易に行えるよう、アスキー形式でプログラムを作成・編集したり、保存したり、中間コード形式 (BASIC) とアスキー形式 (TEXT) の相互変換をしたりできるTEXTモードを設けています。ラムデータファイルのデータもアスキー形式のデータに変換できます。

本章では、このTEXTモードの個々の機能について説明します。

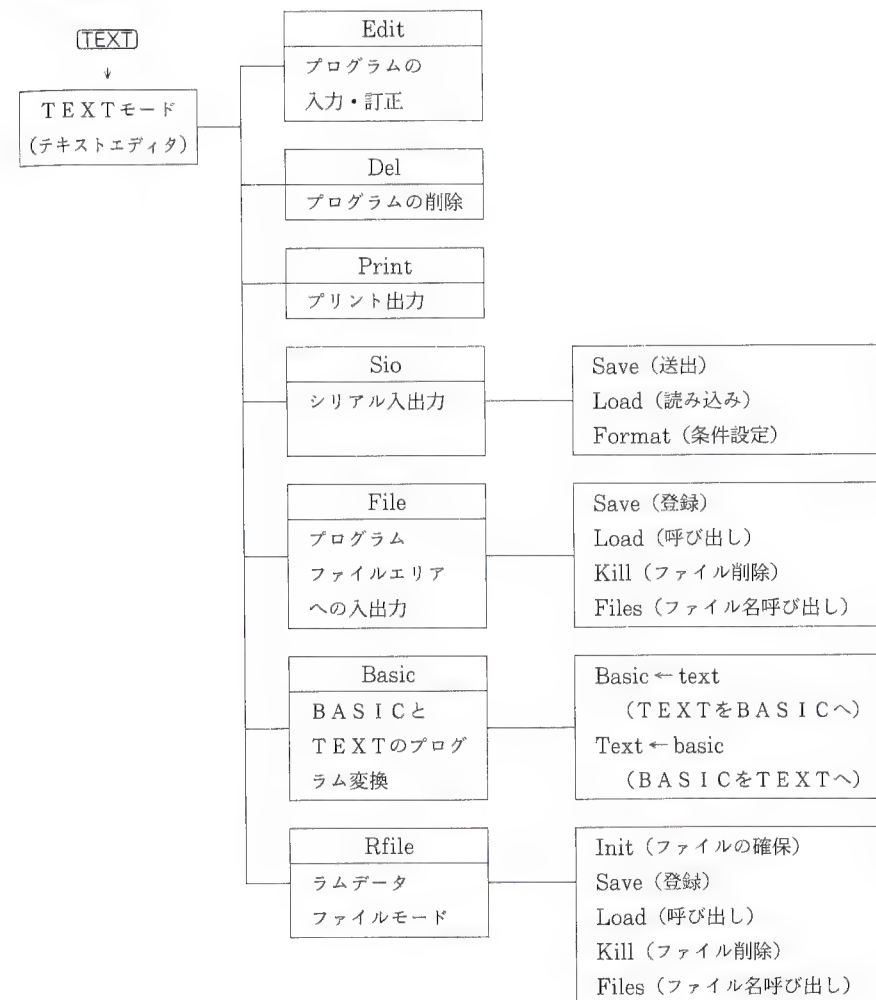
テキストエディタで書き込んだ内容は次の操作でコンパイルもしくはアセンブルします。

- C言語のとき  
(SHIFT) + (TEXT) を押してから (C) (Compile) を押します。
- CASLのとき  
(SHIFT) + (ASMBL) を押してから (C) (Casl) を選んだ後、(A) (Assemble) を押します。
- 機械語プログラム (Z80アセンブラのソースプログラム) のとき  
(SHIFT) + (ASMBL) を押してから (A) (Assembler) を選んだ後、(A) (Asm) を押します。
- PICのとき  
(SHIFT) + (ASMBL) を押してから (P) (Pic) を選んだ後、(A) (Assembler) を押します。

## 1. TEXTモード機能一覧

下記のTEXTモードの機能概略図を参考にしてください。

なお、C言語、CASL、Z80アセンブラおよびPICソースプログラムの入力・編集も、TEXTモードで行います。



## 2. TEXTモードの使いかた

### 2. 1 TEXTモードの設定

RUNモードやPROモードなどで[TEXT]を押すと、右の画面になります。

この画面を機能選択画面と呼びます。

この画面で、表示されている各機能をその頭文字(大文字)に相当するキーを押して選びます。

各機能を選ぶと、それぞれの機能のメニュー画面になったり、機能が働いたりします。

- 各機能が働いている状態で、動作を止めたり、メニュー画面に戻したり、機能選択画面に戻したりするときは、[BREAK]を押します。  
ただし、エラー状態を解除したり、ファイル名など入力中の文字などを消去するときは[CLS]を押してください。
- TEXTモードは、[BASIC]、[SHIFT] + [ASMBL]などのキーでモードを変えたり、電源を切ると解除されます。

```
*** TEXT EDITOR ***
Edit Del Print
Sio File Basic Rfile
```

### 2. 2 エディット機能 (Edit)

機能選択画面で[E]を押せば、エディット機能が選ばれ、エディット画面になります。

[E]

```
TEXT EDITOR
<
```

- エディット機能では、プロンプト記号が“<”になります。(BASICモードでは“>”)
- TEXTプログラムを書き込むときは、BASICプログラムの場合と同じように、行番号を先頭につけて書き込みます。ただし、BASICのように、自動的に行番号の後ろにコロンの(:)をつけたり、命令の後ろにスペースを入れたりしません。入力したとおりに書き込まれます。
- 行の順番は、行番号の順番に並べ替えられます。
- 行番号は、1~65279の範囲でつけることができます。この範囲を超えている場合、または行番号がない場合はエラー(LINE NO. ERROR)になります。エラーは[CLS]で解除します。
- 機能選択画面に戻るときは[BREAK]を押します。

(注) 行番号の次が数字で始まるようなテキスト行を入力することはできません。

数字で始まるような行を入力するときは' (シングルクォーテーション) で行番号と数字を区切って入力してください。

〈例〉 50 ' 100—FORMAT(17X, A) [↵]  
行番号 ↑ シングルクォーテーション

〈例〉 次のプログラムを入力します。

```
10 INPUT A
20 B=A*A
30 PRINT A, B
40 END
```

[キー操作]

```
10 INPUT [SPACE] A [↵]
20 B=A*A [↵]
30 PRINT [SPACE] A, B [↵]
40 END [↵]
```

```
10 INPUT A
20 B=A*A
30 PRINT A, B
40 END
```

#### ご注意

このプログラムはパソコン等へ転送することを考えたプログラムで、CASLのプログラムではありません。そのため、このプログラムをCASLでアセンブルするとエラーになります。CASLの文法については、CASLの説明を参照してください。

### プログラムの編集

TEXTプログラムの変更、修正などはBASICプログラムの変更、修正と同じ方法で行うことができますので参照してください。

なお、次のようにBASIC命令と同等機能の命令が用意されています。それぞれの命令の働きについては、BASICのそれぞれの命令も参照してください。

AUTO命令……………A命令(オート)  
LIST命令……………L命令(リスト)  
RENUM命令……………R命令(リナンバー)  
DELETE命令…………D命令(デリート)  
LCOPY命令……………C命令(コピー)

さらに、TEXTモードでは検索(S命令)・置換(E命令)命令が用意されていますので、編集が容易です。

ただし、TEXTモードでのリナンバーは、行頭の番号だけをつけ直します。

BASICプログラムをTEXTプログラムに変換して、リナンバーを行うとGOTOやTHEN、GOSUB、RESTORE命令などの後の行番号は変更されません。再びBASICプログラムに変換したとき、正しく動作しなくなりますので注意してください。

**A命令書式** A [[開始行] [, 増分]] [↵]

開始行で指定した行番号から、それ以降の行番号を増分で指定した値に従って自動発生します。

**L命令書式** (1) L [↵]

(2) L行番号 [↵]

(3) Lラベル [↵]

**R命令書式** R [新行番号] [, [開始行] [, 増分]] [↵]

開始行で指定したプログラムの行番号を、新行番号に書き換え、それ以降の行番号を増分

で指定した値に従って順次書き換えていきます。

#### D 命令書式 (1) D 開始行番号 [ , [終了行番号] ]

開始行番号から終了行番号までのプログラムを削除します。終了行番号を省略したときは最終行まで削除します。コンマ(,)も含めて省略したときは開始行番号だけを削除します。



(2) D , 終了行番号

最初の行から終了行番号までを削除します。

#### C 命令書式 C コピー元開始行番号, コピー元終了行番号, コピー先開始行番号

コピー元開始行番号からコピー元終了行番号までのプログラム行を、コピー先開始行番号からコピーします。コピー先プログラム行の増分は、コピー元プログラム行の増分と同じです。

#### S 命令書式 S [式,] '文字列'

指定した文字列を検索し、停止します。文字列は16文字まで指定できます。を押すと次の文字列の検索を行い、を押すと検索を終了します。また、最終行(または最初の行)まで検索を行っても終了します。

式で検索の方向を指定します。

1: 最初の行より行番号の大きい方向に検索します。

0: 最終行より行番号の小さい方向に検索します。


省略した場合は“1”の指定になります。


(注) ・行番号は検索できません。

・'(ダブルクォーテーション)を検索するときは、文字列として $\backslash$ 'を指定し、 $\backslash$ 'を検索するときは $\backslash\backslash$ を指定します。 例: S 1 $\backslash$ ' $\backslash$

#### E 命令書式 E [式,] "文字列1", "文字列2"

文字列1を検索し、停止します。文字列1、文字列2は16文字まで指定できます。

を押すと文字列1を文字列2に置換(置き換え)します。

を押すと置換を行わず次の文字列の検索・置換に進みます。

を押すと検索・置換を終了します。

式で検索の方向を指定します。

1: 最初の行より行番号の大きい方向に検索します。

0: 最終行より行番号の小さい方向に検索します。

省略した場合は“1”の指定になります。


(注) ・置換することにより1行が255文字を超えると、“STRING TOO LONG”のエラーが表示されます。

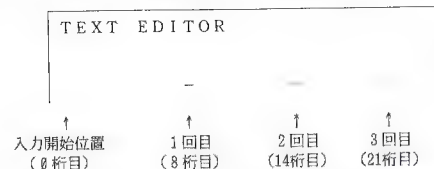
・行番号は検索できません。


・'(ダブルクォーテーション)を検索するときは、文字列として $\backslash$ 'を指定し、 $\backslash$ 'を検索するときは $\backslash\backslash$ を指定します。 例: E 0, 1 $\backslash$ ' $\backslash$  'A'

( [ ] 内の指定は省略可です。)


### の働き

エディット機能の中での  は、次の図に示すようにカーソルを送ります。




最初  を押すとカーソルが8桁進み、2回目を押すと6桁進みます。3回目以降は7桁ずつ進みます。


## 2.3 TEXTプログラムの消去 (Delete)


機能選択画面で  を押せば、消去確認画面になります。



```
*** TEXT EDITOR ***
TEXT DELETE OK? (Y)
```

 を押せば、TEXTプログラムなど、テキストエリアの内容が消去され、機能選択画面に戻ります。

 以外のキーを押すと、消去されずに機能選択画面に戻ります。

● TEXTモードでの記憶内容がない場合は、機能選択画面で  を押しても何も実行されません。


## 2.4 TEXTプログラムの印字 (Print)


プリンタC E-126Pをお持ちの場合は、接続して電源を入れ、機能選択画面で  を押せば印字します。




```
*** TEXT EDITOR ***
--- PRINTING ---
```

印字が終われば、機能選択画面に戻ります。

● 印字を途中で止めるときは  を押してください。

● プリンタの電源が入っていないときや、プリンタが接続されていないときは、機能選択画面で  を押しても何も実行されません。

## 2.5 シリアル入出力 (Sio)

機能選択画面で  を押せば、シリアル入出力のメニュー画面になります。



```
<< SIO >>
Save Load Format
```

この画面で、出力(Save)、読み込み(Load)、条件設定(Format)を選びます。それぞれの機能の頭文字に相当するキーを押します。

● 通信を行う前に、入出力条件を通信する相手側と合わせておく必要があります。

### ①入出力の条件設定 (Format)

通信を行う場合の入出力の条件を設定します。



シリアル入出力のメニュー画面で[F]を押すと、次の画面になります。しばらく待つか、いずれかのキーを押すと、条件設定画面になります。

[F]

```
<< SIO >>
Select ←, →, ↑, ↓ key
Set      ← key
- push any key
```

```
→ baud rate = 1200
data bit    = 8
stop bit    = 1
parity      = none
end of line = CR LF
end of file = 1A
```

(条件設定画面)

→マークは変更できる項目を示しています。変更したい項目を[▼]、[▲]で移動させて選択します。条件は8種類あり、[▼]を押していくと、隠れている部分が1行ずつ表示されます。

⋮  
[▼]

```
stop bit    = 1
parity      = none
end of line = CR LF
end of file = 1A
line number = yes
→ flow      = RS/CS
```

初期状態では以上の値が設定されています。

項目を選択し、[▶]または[◀]で条件(値)を切り替えます。ただし、end of file(エンド オブ ファイル)は直接コードを入力します。

そして[↵]を押せば設定されます。[↵]を押さないと、前の条件のままになります。

### 【条件の説明】

- 通信速度 : 300、600、1200、2400、4800、9600  
(baud rate) データ転送速度の指定で、本機では上記の速度(bps)が指定できます。  
bps: bit per second (1bps=1ビット/秒)
- ワード長\* : 7、8  
(data bit) 1文字を何ビット構成で送受信するかを指定します。
- ストップビット数\* : 1、2  
(stop bit) ストップビット数を1ビットにするか2ビットにするかを指定します。
- パリティ : none、even、odd  
(parity) パリティビットをどのように扱うかを指定します。  
none …パリティビットの送受信を行いません。  
even …偶数パリティが指定されます。  
odd …奇数パリティが指定されます。
- 区切りコード : CR、LF、CR LF  
(end of line) プログラムライン(行)の終了を示す区切りコードを指定します。  
CR ……CR(キャリッジリターン)コードが指定されます。  
LF ……LF(ラインフィード)コードが指定されます。

CR LF…CRコード+LFコードが指定されます。

(注)ここで指定するコードはテキストプログラムの行末を示すコードです。

PRINT#やfprintfで出力されるデータの区切りコードは、常にCR LFが指定されます。

- テキスト終了コード: 00~FF(2桁の16進数値)  
(end of file) プログラムなどの終了を示すテキスト終了コードを指定します。
- 行番号つき入出力 : yes、no  
(line number) ● 出力時  
yes ……行番号をつけて出力します。  
no ……行番号を省いて出力します。
- 読み込み時  
yes ……行番号を付加しません。  
読み込むプログラムに行番号がついているときに指定します。  
読み込んだプログラムに行番号がついていないと、エラー(LINE NO. ERROR)になります。  
no ……自動的に行番号(10から10きざみ)をつけます。
- フロー制御\* : RS/CS、Xon/Xoff、none  
(flow) 通信を行うときの制御方法を指定します。  
RS/CS ……RS/CS信号で制御します。  
Xon/Xoff ……Xon/Xoffコードで制御します。  
none ……制御を行いません。

※OPEN "COM1:"およびfopen("stdaux1",として全二重通信を指定しているときは、常にワード長は8ビット、ストップビット数は1ビットになります。これ以外の場合は、常にフロー制御はRS/CSに固定されます。

入出力条件は前に述べた方法で変更できます。変更した入出力条件はリセットスイッチを押してメモリを消去するか、電池交換を行うか、条件の変更を行うまで保持されます。

### ②出力(Save)

シリアル入出力のメニュー画面で,[S]を押すとシリアル入出力端子への出力が開始されます。

[S]

```
<< SIO >>
--- SENDING ---
```

出力が終了すれば、シリアル入出力のメニュー画面に戻ります。

- 出力を中断するときは、[BREAK]を押します。メニュー画面に戻ります。
- TEXTモードでの記憶内容がない場合は,[S]を押しても何も行われません。

### ③読み込み(Load)

シリアル入出力のメニュー画面で,[L]を押すとシリアル入出力端子へ送られてくるデータの読み込みが開始されます。



&lt;&lt; SIO &gt;&gt;

--- RECEIVING ---

読み込みが終了（テキスト終了コードを受信）すれば、シリアル入出力のメニュー画面に戻ります。

- 読み込みを中断するときは **BREAK** を押します。メニュー画面に戻ります。
- データが正常に読み込めない場合や、パリティチェックで異常が発生した場合などではエラー（I/O DEVICE ERROR）になります。**CLS** でエラーを解除してください。

## 2. 6 ミニI/OからのTEXTプログラム送出

TEXTモードの機能選択画面で **L** を押すと、ミニI/Oの平行ポートからTEXTプログラムを送出します。

詳しくは、先生の指導に従ってください。（366ページを参照）

## 2. 7 プログラムファイル（File）

機能選択画面で **F** を押すと、プログラムファイルのメニュー画面になります。



&lt;&lt; PROGRAM FILE &gt;&gt;

Save Load Kill Files

この画面で、登録（Save）、呼び出し（Load）、削除（Kill）およびファイル名の確認（Files）を選びます。それぞれの頭文字に相当するキーを押します。

- 登録したファイルはプログラムファイルエリアに書き込まれます。（158ページ参照）

### ①TEXTプログラムの登録（Save）

プログラムファイルのメニュー画面で **S** を押すと、ファイル名の入力待ちになります。



&lt;&lt; PROGRAM FILE &gt;&gt;

→Save Load Kill Files

FILE NAME=?

ファイル名を入力して **↵** を押せば、登録が行われます。

〈例〉ファイル名を“TEST”とします。

TEST



&lt;&lt; PROGRAM FILE &gt;&gt;

→Save Load Kill Files

FILE NAME=TEST\_

登録が行われ、プログラムファイルのメニュー画面に戻ります。

入力したファイル名がすでに登録されているときは、すでに登録されているファイルの内容を新しい内容に書きかえてよいかどうかを聞いてきます。（FILE OVERWRITE OK? (Y) と表示）

**Y** を押せばファイルが登録され、プログラムファイルのメニュー画面に戻ります。

**Y** 以外のキー（そのとき有効に働くキー）を押せば、登録は中止されます。

- ファイル名を省略することはできません。ファイル名を入れずに **↵** を押すとエラー（ILLEGAL FILE NAME）になります。
- ファイル名は8文字以下の名前と、3文字以下の拡張子を指定できます。  
拡張子をつけなかった場合は、自動的に“.TXT”がつけられます。
- 空きエリアの容量が足りない場合は、エラー（MEMORY OVER）になります。  
空きエリアには、ファイルサイズ+34バイト（ファイル名の記憶などに使用）以上が必要です。
- テキストエリアにTEXTプログラムがない場合は、登録は行われません。

### ②TEXTプログラムの呼び出し（Load）

プログラムファイルのメニュー画面で **L** を押すと、登録されているプログラムファイル名を表示し、最初のファイル名の左側に“LOAD →”と表示されます。（何も登録されていないときは、画面は変わりません。）



（画面は、すでにこれらのプログラムが登録されている場合の例です。）

|           |      |      |
|-----------|------|------|
| LOAD →ABC | .TXT | 456  |
| PRO       | .TXT | 1234 |
| サンプ ル 01  | .BAS | 1567 |
| TEST      | .TXT | 789  |

“LOAD →”表示を **▼**、**▲** で呼び出したいファイル名の前に移し、**↵** を押せば、そのファイルの内容が呼び出され、メニュー画面に戻ります。表示されているファイル名以外にファイルがある場合は、**▼**、**▲** で移動させていくことにより1行ずつ表示されます。

- TEXTモードで呼び出せるプログラムは、TEXTモードで登録したプログラムだけです。BASICプログラム（BASICのSAVE命令で登録したプログラム）を呼び出そうとするとエラー（FILE MODE ERROR）になります。

### ③ファイルの削除（Kill）

プログラムファイルエリアに登録されているプログラム（ファイル）を削除します。

プログラムファイルのメニュー画面で **K** を押すと、ファイル名の入力待ちになります。（何も登録されていないときは、画面は変わりません。）



&lt;&lt; PROGRAM FILE &gt;&gt;

Save Load →Kill Files

FILE NAME=?

削除したいプログラムのファイル名を入力して **↵** を押します。

〈例〉ファイル名“TEST”のプログラムを削除します。

TEST

```
Save Load →Kill Files
FILE NAME=TEST_
```



ファイルの削除確認画面になります。

(FILE DELETE OK? (Y) と表示されます。)

☒ を押せばファイルが削除され、プログラムファイルのメニュー画面に戻ります。

☐ 以外のキー（そのとき有効に働くキー）を押せば、削除は中止されます。

● 拡張子の指定を省略した場合は、“.TXT”の指定とみなされます。

● 指定したファイル名が見つからない場合はエラー (FILE NOT FOUND) になります。

#### ④ファイル名とファイルサイズの確認 (Files)

プログラムファイルのメニュー画面で ☐ を押すと、登録されているファイル名とファイルサイズが表示され、最初のファイル名の左側に → マークが表示されます。(何も登録されていないときは、画面は変わりません。)



(画面は、すでにこれらのプログラムが登録されている場合の例です。)

```
→ABC .TXT 456
PRO .TXT 1234
サンプル001.BAS 1567
TEST .TXT 789
```

表示されているファイル名以外にファイルがある場合は、☐、☐ で → マークを移動させていくことにより1行ずつ表示されます。

● このファイル名の確認画面で、☐ + ☐ (または ☐ + ☐ ) と操作すると、→マークで示しているファイル名のプログラムを呼び出すことができます。

#### ファイルサイズについて

● テキストファイルのファイルサイズは、各行のバイト数の合計です。各行のバイト数は、3バイト (ラインナンバー) + テキストのバイト数 + 1バイト (改行) で計算されます。

〈例〉10→INPUT→A ☐

ライン      テキスト      改行

この行のバイト数は、3 + 8 + 1 = 12バイトです。

● RUNモードやPROGRAMモードで登録したBASICプログラムのファイルサイズは中間コードに変換後のサイズです。(RUNモードやPROGRAMモードで登録したBASICプログラムは中間コードに変換して登録されています。)

## 2. 8 BASICコンバータ (Basic)

BASICプログラム (中間コード形式) をTEXTプログラム (アスキー形式) に変換したり、逆にTEXTプログラムをBASICプログラムに変換したりする機能です。本機用のBASICプログラムをパソコン等で管理する場合などに利用します。

機能選択画面で ☐ を押すと、BASICコンバータのメニュー画面になります。



```
<< BASIC CONVERTER >>
Basic←text Text←basic
```

この画面で、BASICへの変換 (Basic ← text)、TEXTへの変換 (Text ← basic) を選ぶことができます。それぞれの頭文字に相当するキーを押します。

#### ①TEXTとBASICのプログラム変換 (アスキー形式と中間コード形式の変換)

メニュー画面で、☐ を押せばTEXTプログラムをBASICプログラムに変換し、プログラム・データエリアに入れます。

☐ を押せばBASICプログラムをTEXTプログラムに変換し、TEXTエリアに入れます。

〈例〉TEXTプログラムをBASICプログラムに変換します。



```
<< BASIC CONVERTER >>
--- CONVERTING ---
```

変換を実行します。その後、機能選択画面に戻ります。

(変換する内容が少ない場合、この画面の表示は一瞬で終わります。)

● BASICプログラムに変換するとき、BASICのプログラム・データエリアにBASICプログラムがあった場合、あるいはTEXTプログラムに変換するとき、テキストエリアにTEXTプログラムがあった場合は、以前のプログラムを削除してよいか聞いてきます。



```
→Basic←text Text←basic
BASIC DELETE OK? (Y)
```

☐ を押せば、以前のプログラムが削除され、変換が開始されます。

☐ 以外のキー（そのとき有効に働くキー）を押せば変換は中止され、機能選択画面に戻ります。

● 通常、変換した場合でも変換元のプログラムは保持していますが、メモリ容量が足りない場合は、次のように、変換元のプログラムを削除してよいか聞いてきます。

```
--- CONVERTING ---
TEXT DELETE OK? (Y)
```

☐ を押せば、変換を行いながら変換元のプログラムを消去していきます。変換が終われば、変換元のプログラムはすべて消去されます。

☐ 以外のキー（そのとき有効に働くキー）を押せば変換は中止され、機能選択画面に戻ります。

## 2. 9 ラムデータファイル (Rfile)

シーケンシャルデータをラムデータファイルエリアに登録し、BASICプログラムなどで任意に読み出して使用することができます。

機能選択画面で ☐ を押すと、ラムデータファイルのメニュー画面になります。

```
<< RAM DATA FILE >>

Init   Save   Load   Kill
Files
```

このメニュー画面で、ファイルの確保 (Init)、テキストエリアからの登録 (Save)、テキストエリアへの呼び出し (Load)、ファイルの削除 (Kill) およびファイル名の確認 (Files) を選びます。それぞれの頭文字に相当するキーを押します。

### ①ファイルの確保 (Init)

ラムデータファイルを使用するときは、あらかじめ Init 機能でファイルの名前と大きさを確保しておかなければなりません。

ラムデータファイルのメニュー画面で  を押すと、ファイル名の入力待ちになります。

```
→Init   Save   Load   Kill
Files

FILE NAME=?
```

〈例〉ファイル名を “TEST” とします。

TEST. DAT

```
→Init   Save   Load   Kill
Files

FILE SIZE=?
```

次にファイルの大きさをバイト単位で指定します。このファイルには、指定した容量の範囲内でデータを登録することができます。

ここでは 1024 バイトの容量にします。

1024

```
→Init   Save   Load   Kill
Files

FILE SIZE=1024_
```

入力したファイル名がすでに登録されているときは、すでに登録されているファイルを消去して、新しいファイルとして確保してよいかどうかを聞いてきます。(FILE INITIALIZE OK? (Y) と表示されます。)

☒ を押せば、新しいファイルを確保します。

☒ 以外のキー (そのとき有効に働くキー) を押せば、ラムデータファイルのメニュー画面に戻ります。

〔補足〕 ● 拡張子を省略した場合は、自動的に “. DAT” がつけられます。

- ファイル名は 8 文字以内の文字で指定します。
- 指定した容量以上のデータは登録できませんので、少し大きめの容量を確保しておくことをおすすめします。ただし、あまり容量を大きくすると、フリーエリア (メモリの未使用部分) が少なくなります。なお、ファイルの最後に終了コードを書き込みますので、実際に書き込めるバイト数は (指定した容量 - 1) バイトです。
- ファイル名などを記憶する制御用として、34 バイト使用します。指定する容量 (バイト) + 34

バイト以上のフリーエリアがないと “MEMORY OVER” エラーになります。なお、容量として 0 バイトを指定すると無視されます。

### ②テキストエリアからの登録 (Save)

テキストエリアに書き込まれているプログラム (データ) を、データとしてラムデータファイルに登録します。登録するとき行番号は省略して登録します。

ラムデータファイルのメニュー画面で  を押すと、ファイル名の入力待ちになります。

```
<< RAM DATA FILE >>

Init   →Save   Load   Kill
Files

FILE NAME=?
```

〈例〉ファイル名を “TEST” とします。

TEST

```
Init   →Save   Load   Kill
Files

FILE NAME=TEST_
```

入力したファイルにデータがすでに登録されているときは、すでに登録されているデータを新しいデータに書きかえてよいかどうかを聞いてきます。

(FILE OVERWRITE OK? (Y) と表示されます。)

☒ を押せばファイルが登録され、ラムデータファイルのメニュー画面に戻ります。

☒ 以外のキー (そのとき有効に働くキー) を押せば、登録は中止されます。

- 登録が行われメニュー画面に戻ります。
- 拡張子の指定を省略した場合は、 “. DAT” の指定とみなされます。
- 指定したファイル名がなかったり、容量が足りなかった場合は、エラーになります。
- TEXTモードでの記憶内容がない場合は、 を押しても何も実行されずにメニュー画面に戻ります。
- ラムデータファイルへの登録 (書き込み) は BASIC プログラムの PRINT # 命令で行うこともできます。

### ③テキストエリアへの呼び出し (Load)

ラムデータファイルに登録されているデータを、テキストエリアに呼び出しします。呼び出しするとき行番号は自動的に 10 番ごとに付加します。

ラムデータファイルのメニュー画面で  を押すと、登録されているファイル名と確保したサイズを表示し、最初のファイル名の左側に “LOAD →” と表示されます。(何も登録されていないときは、画面は変わりません。)

(画面は、すでにこれらのファイルが登録されている場合の例です。)

```
LOAD →TEST   . DAT   1024
      ABC     . DAT    512
      SAMPLE  . DAT   2048
```

“LOAD →” 表示を 、 で呼び出ししたいファイル名の前に移し、 を押すと、そのファイルの内容がテキストエリアに行番号付きで呼び出しされ、メニュー画面に戻ります。

表示されているファイル名以外にファイルがある場合は、、 で移動させていくことにより 1 行ず



つ表示されます。

#### ④ファイルの削除 (Kill)

ラムデータファイルエリアに登録されているデータファイルを削除します。

ラムデータファイルのメニュー画面で **[K]** を押すと、ファイル名の入力待ちになります。(何も登録されていないときは、画面は変わりません。)

**[K]**

```
<< RAM DATA FILE >>
Init   Save   Load  ➔Kill
Files
FILE NAME=?
```

削除したいデータファイルのファイル名を入力して **[↵]** を押すと、削除が行われます。

〈例〉ファイル名 “TEST” のファイルを削除します。

TEST **[↵]**

ファイルの削除確認画面になります。(FILE DELETE OK? (Y) と表示されます。)

**[Y]** を押せば削除が行われ、メニュー画面に戻ります。

**[Y]** 以外のキー (そのとき有効に働くキー) を押せば、削除は中止されます。

● 拡張子の指定を省略した場合は、“. DAT” の指定とみなされます。

● 指定したファイル名が見つからない場合はエラーになります。

#### ⑤ファイル名と確保したサイズの確認 (Files)

ラムデータファイルのメニュー画面で **[F]** を押すと、登録されているファイル名と確保したサイズを表示し、最初のファイル名の左側に ➔ マークが表示されます。(何も登録されていないときは、画面は変わりません。)

**[F]**

(画面は、すでにこれらのファイルが登録されている場合の例です。)

```
➔TEST      . DAT   1024
ABC         . DAT   512
SAMPLE     . DAT   2048
```

表示されているファイル名以外にファイルがある場合は、**[▼]**、**[▲]** で ➔ マークを移動させていくことにより1行ずつ表示されます。

● このファイル名の確認画面で、**(SHIFT) + (M)** (または **(2nd F) (M)**) と操作すると、➔ マークで示しているファイル名のデータをテキストエリアに呼び出しできます。

#### 【まとめ】

次の手順のように、BASICプログラムで書き込んだラムデータファイルのデータをパソコン等へ送出できます。なお、パソコンと接続するためには、ケーブルが必要です。(371ページ付録参照)

- ① **(TEXT) (R) [ ]** でファイルを確保
- ② BASICでデータを登録 (書き込み)
- ③ **(TEXT) (R) [L]** でテキストエリアに呼び出し
- ④ **(TEXT) [E]** で内容の確認

⑤ **(TEXT) (S) [F]** でフォーマット設定

⑥ **(TEXT) (S) [S]** でパソコン等へ出力

(注) ● BASICのPRINT # 命令でデータを書き込んだときのデータのフォーマットについては、PRINT # 命令を参照してください。CR、LFコードはテキストプログラムでの行の区切りになります。

なお、データの最初が数字の文字データ (例 "123") の場合は、TEXTモードでは行番号との区別がつかず、正しく編集できません。

#### ご注意

● 変換元のプログラムを消去しながら変換していても、なおメモリが足りなくなるとエラー (MEMORY OVER) になります。

この場合、変換された部分と、変換できなかった部分がTEXTとBASICに分かれてしまいます。したがって、変換する前にプログラムをプリンタに印字しておくことをお勧めします。特にBASICからTEXTに変換するときに、起こる可能性が高いので注意してください。

● パスワード (PASS命令参照) が設定されているときは、BASICコンバータに入れません。あらかじめ、RUNまたはPROモードでパスワードを解除してください。

● BASICコンバータでは、データエリア (変数) を消去しません。フリーエリア (空きエリア) の容量が少ないと、変換できなくなることがあります。あらかじめCLEAR命令で消去するなど、フリーエリアを広げておいてください。

● TEXTからBASICに変換するときは、TEXTの内容が何であっても、それを本機用のBASICプログラムと見なして変換作業を行います。したがって、TEXTの内容によってはBASICに変換して、再度TEXTに変換しても、元の内容に戻らない場合があります。

```
〈例〉 TEXT      10 FORMULA
      ↓ (変換)
      BASIC     10 : FORMULA
      ↓ (変換)
      TEXT      10 FORMULA
```

# 第6章

## C言語機能

C言語はUNIX\*というオペレーティングシステムを記述するために生まれました。しかし、今ではパソコンはもちろん、ワークステーションやミニコンでのポピュラーなプログラミング言語として定着しています。

C言語は、基本的な部分の統一性が高いため、異機種間でのプログラミングの移植性がたいへん高く、本機で開発したプログラムであっても、少しの手直しでパソコンなどで実行できるという特徴があります。

本章では、C言語で書かれたプログラムを本機で実行する方法および一般的なC言語との違いについて説明します。

C言語自身については、入門書等の関連書籍が豊富に市販されていますので、それらの書籍を参照してください。

※UNIXオペレーティングシステムはUNIX System Laboratories, Inc. が開発しライセンスしています。

## 1. C言語の特徴

C言語の特徴は、言語自体がコンパクトでありながら、ハードウェアの制御やシステムプログラムからアプリケーションプログラムまで扱える守備範囲の広さにあります。

### ■ハードウェアに密着したプログラムが作れる

BASICやFORTRANのような高級言語でありながら、ビット演算や、メモリのアドレスが直接扱えます。これらの機能により、これまでアセンブラ言語に頼ってきたハードウェアの操作や外部機器の制御が、C言語によって記述できるようになりました。そのため、これらのプログラムの開発が効率的に行えるようになりました。

### ■高級言語としての機能を備えている

構造化プログラミングによる、読みやすく、わかりやすいプログラムの記述ができます。そのため、プログラムの開発が効率的に行えます。  
また、数値処理やデータ処理のためのデータ型が豊富に用意されています。そのため、科学計算プログラムやアプリケーションプログラムなど、幅広い目的のプログラミングが可能です。

### ■コンパクトな記述ができる

プログラムをコンパクトに記述するための機能（演算子）が豊富にそろっています。また、ポインタ操作によって実行効率の高いプログラムが作れます。

### ■移植性が高い

ハードウェアに密着したプログラムの移植性が高いことがC言語の流行の大きな要因です。異機種間でのCプログラムの移植性も高く、本機で開発したプログラムであっても、少しの手直しでパソコンなどで実行できるようになります。

このように大変強力なC言語ではありますが、少々欠点もあります。

それは、機能が豊富なため難解なプログラムができたり、システム自体が暴走するプログラムが記述できるという危険性があることです。

#### ご注意

したがって、機械語プログラムと同様、消えては困るプログラムやデータは、紙に書き写しておいてください。CE-126P（プリンタ）をお持ちの場合は、印字しておいてください。

## 2. 始めようCプログラミング

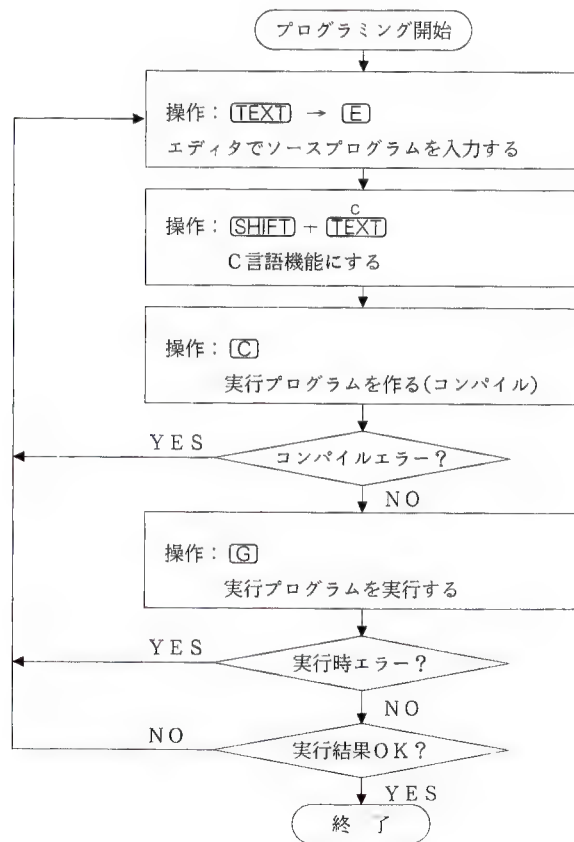
ここでは本機でのプログラミングの方法と、Cプログラムの基本的な約束ごとを説明します。

### 2. 1 Cプログラムの作成から実行まで

C言語などのプログラムは人間が理解しやすい言葉、約束ごと（文法）で記述されます。しかし、C言語のままではコンピュータは直接理解できません。

そのため、C言語で記述されたプログラムは、コンピュータが理解できる機械語に変換する必要があります。

したがって、本機ではソースプログラムの作成（C言語でのプログラミング）、コンパイル（変換）、実行という手順でプログラミングします。



#### ■ステップ1：ソースプログラムを作る

メッセージを出力する簡単なプログラムを作ってみましょう。最初にC言語でソースプログラムを作ります。

#### ◆テキストエディタを呼び出す

TEXT（テキスト）モードに入り、エディタ機能呼び出します。

TEXT → E

TEXT EDITOR  
<

#### ◆ソースプログラムを入力する

次のプログラムを入力します。

```

1 0 __main ()
2 0 {
3 0 __printf ('C プログラミング %n');
4 0 }
```

CAPS を押して小文字入力モードにします（“CAPS”を消灯させます）。

行の先頭に行番号を入力してから各行のプログラムをキー入力してください。カナ入力は、カナを押してからローマ字で入力します。英字入力に戻るにはもう一度カナを押します。

各行の終わりでは必ずEnterを押してください。行番号30は画面上では1行に収まらず2行に表示されますが、Enterが押されるまでは1行として扱われます。—はSPACEの入力を意味します。

もし、入力をまちがえた場合などは、「第5章 TEXTモード」を参照して修正してください。

一般のプログラムでは行番号をつけるとエラーになります。しかし、本機のテキストエディタは行番号をつけて入力しなければならないため、このような記述となっています。ほかのC言語処理系では、行番号は取り除かなければなりません。

#### ■ご注意

この項（2. 始めようCプログラミング）では、プログラム中にスペースマークやEnterを入れています。

#### ■ステップ2：ソースプログラムをコンパイルする

ソースプログラムをコンパイルし実行プログラムを作ります。

#### ◆C言語機能のメニュー画面にする

SHIFTを押しながらTEXTを押すと、C言語機能のメニュー画面が呼び出されます。

SHIFT + TEXT

\*\*\* C \*\*\*  
Compile Trace Go Stdout

各コマンドは次のような働きをします。

Compile：TEXTで作成したプログラムをコンパイルする

Trace：コンパイルしたプログラムをトレース実行する

Go：コンパイルしたプログラムをノーマル実行する

Stdout：出力を画面 ↔ プリンタとで切り替える

（プリンタ動作可能時、Sを押すたびに切り替わる）

各コマンドはコマンドメニューの先頭文字を入力することで実行されます。

## ◆コンパイルする

**[C]**を押してください。コンパイルが実行されます。

**[C]**

コンパイル実行中は **compiling** と一時表示し、次の画面に移ります。ただし、プログラムが短いときは、この表示は一瞬で終わります。

## ◆コンパイルが終了すると

コンパイルが終了すると次の画面が表示されます。

```
*** C ***
Compile Trace Go Stdout
complete !
```

プログラムに悪いところがあると、エラーのある行番号とエラーの内容が表示され、コンパイルが中断されます。このような場合は **[TEXT]** **[E]** と押してテキストエディタに戻り、プログラムを修正してからもう一度コンパイルしてください。

(注) memory full と表示されたときは、コンパイルに必要なメモリが足りません。不要になった BASIC の変数やプログラムを消去してから、もう一度コンパイルしてください。

## ■ステップ3：プログラムを実行する

プログラムの実行にはノーマル実行と、トレース実行の2つのモード（機能）があります。ここではノーマル実行を行います。またどちらのモードも実行プログラムが作成されていないときは、自動的にコンパイルをして実行プログラムを作り、実行します。

トレース実行については「2.2 プログラムのトレース実行」で説明します。

## ◆ノーマル実行する

コンパイルが終了したら **[G]**を押してください。プログラムが実行され、画面には次のように表示されます。

**[G]**

```
C プログラムのラング
*EXIT (40)
```

\*EXIT (40) はプログラムの実行が終了した行番号を示しています。**[CLS]**、**[F5]**、**[BREAK]** でメニューに戻ります。

## ◆実行時エラーが発生したら

実行時にエラーが見つかることがあります。実行時エラーが発生すると、エラーメッセージを出力して実行を停止します。コンパイルエラーと同様に、エディタを呼び出してソースプログラムを修正してください。

実行時エラーメッセージについては243ページを参照してください。

## 2.2 プログラムのトレース実行

プログラムの構造が複雑になると、エラーが発生するときや、期待どおりの結果が得られないときの処理が大変になります。エラーを発見するために便利なのがトレース実行です。

トレース機能とはプログラムを1ステップずつ実行していく機能のことです。

各ステップごとに変数の値を調べ、エラーの原因を探していきます。

ここでは、次のプログラムを例に、トレース実行します。テキストエディタで入力して、C言語機能でコンパイルしてください。

```
1 0 _main ()
2 0 _ {
3 0 _int i, gokei = 0;
4 0 _for (i = 1; i < 51; i++) _ {
5 0 _ _gokei += i;
6 0 _ _printf ('1 + . . . + %d = %d ¥ n', i, gokei);
7 0 _}
8 0 _}
```

## ■トレースモード

C言語機能のメニュー画面で **[F5]** を押し、トレースモードにします。

画面には次のように、最初に実行する行が表示されます。

```
? 1 0 _main ()
```

**[F5]** を押せば、? の行を実行し、次のステップへ進みます。順次 **[F5]** で実行を進めます。

**[BREAK]** を押せば、ブレークモードに入ります。

## ■ブレークモード

ブレークモードでは次のキー操作ができます。

**[F5]** : 実行を再開します。

**[C]** : 実行を再開します。

**[A]** : 実行を中断してメニュー画面に戻ります。

**[T]** : トレースを指定して、実行を再開します。

**[N]** : トレースを解除して、実行を再開します。

**[D]** : 変数表示モードに入ります。

変数表示モードでは変数名を入力すると、そのステップでの変数の内容が表示されます。変数 i の値を表示させてみましょう。

**[BREAK]**

ブレークモードになります。

**[D] i**

```
1 + . . . + 1 = 1
4 0 _for (i = 1; i < 51; i++) {
Break > D
ヘンソウ> i
int : 2 (0x0002)
ヘンソウ> _
```

i の値が2の場合



- [補足] ・ **[T]** を押してトレースモードにした後、**[↩]** を5回押すと **i** はカウントアップされ2になります。
- ・ **i** は必ず小文字 (“CAPS” 消灯) で入力してください。プログラムでの指定が小文字のためです。**[D]** を押すときは “CAPS” が消灯していても有効です。

画面には **i** のデータ型 (int) とその値が表示されます (カッコ内は16進表示で前に **0x** がつきます)。変数表示モードは **[BREAK]** を押せば終了し、ブレークモードに戻ります。

## 2. 3 表示出力と印字出力の切り替え

C E-126 P (プリンタ) をお持ちの場合は、本機とプリンタが接続され、動作可能状態になっているとき、C言語機能のメニュー画面で **[S]** を押すと、画面の **Stdout** が **Stdprn** に切り替わります。

もう一度 **[S]** を押すと **Stdout** に戻ります。

この切り替えは、下記の出力関数で、出力 stream が stdout に指定されている場合に、出力先を画面とプリンタとで切り替えるときに使用します。

出力 stream が stdprn に指定されているときは、メニュー画面での指定にかかわらず、プリンタに出力されます。

**Stdout 表示時**: 画面に出力

**Stdprn 表示時**: プリンタに出力

C言語機能選択時などには Stdout になります。

〈有効な関数〉

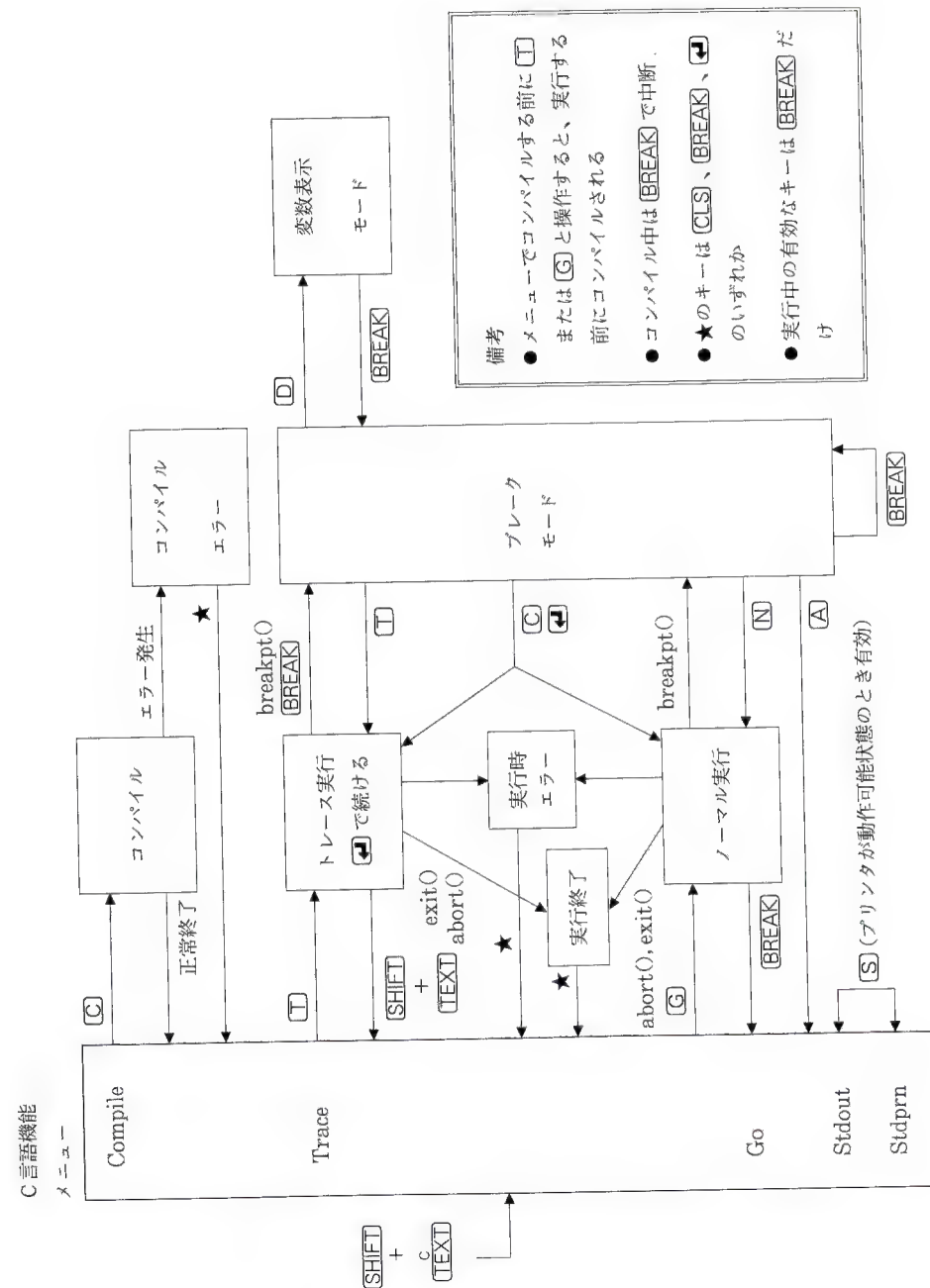
putc  
fputc  
fputs  
fprintf

### ご注意

Cプログラム実行中は、電池が消耗しても “**BATT**” が点灯しませんので、実行させたまま長時間放置しないでください。実行させたまま長時間放置しますと、電圧が低下し、正常な動作をしなくなる恐れがあります。

なお、実行中に誤動作等が発生した場合は電池の消耗が考えられます。**[BERAK]** またはリセットスイッチを押して実行を止め、“**BATT**” の点灯または電源が切れる場合は、速やかに電池を交換してください。

## 2. 4 C言語機能一覧



## 2. 5 Cプログラムのスタイル

BASICになれている人は、C言語のプログラムはとても雰囲気が違うと感じているかもしれません。その理由のひとつは、C言語のプログラムは小文字を中心に記述されることにあります。Cプログラムでは特別な場合を除いて大文字は使いません。

そのほかにもC言語特有のプログラミングスタイルがあります。ここではそのうちの基本的な約束ごとを説明します。

テキストエディタを呼び出して、次のプログラムを入力してください。

```
1 0 _main ()
2 0 _ {
3 0 _ printf ( ' タノシイ _ ' ) ;
4 0 _ printf ( ' C _ プ ロ グ ラ ミ ン グ * ¥ n ' ) ;
5 0 _ }
```

### ■基本的なスタイル

Cプログラムの最も基本的な構成は次のようになります。

```
main ()
{
    実行文
}
```

mainは文字どおり、Cプログラムのメイン（主プログラム）であることを示しており、プログラムの実行は、必ずmain () から始まります。main () で実行する内容は { と } との間に書きます。これがC言語の基本的な約束で、プログラムの実行の始まりと終わりがはっきりとわかるような形式になっています。

入力したプログラムで実行されるのは、次の2つの出力文です。

```
printf ( ' タノシイ _ ' ) ;
printf ( ' C _ プ ロ グ ラ ミ ン グ * ¥ n ' ) ;
```

printf文は「」で囲まれた文字の並び（文字列）を出力する命令（C言語では関数という）です。

### ■実行の区切り

それぞれの実行文の終わりに ;（セミコロン）がついていることに注目してください。;は一つの実行単位（文）の終わりを示す大切な目印です。初心者は ; を忘れがちです。気をつけてください。  
; を忘れると、次の行以降でエラーになります。

BASICでは行が実行の単位（区切り）を意味していましたが、Cプログラムは行という単位にしばられないで記述できます。これをフリーフォーマットといいます。

先のプログラムは次のように書くことができます。

```
main () { printf ( " タノシイ _ " ) ; printf ( ' C _ プ ロ グ ラ ミ ン グ * ¥ n ' ) ; }
```

同じ働きをするプログラムでも、書きかたによって読みやすさ、わかりやすさが違ってきます。フリーフォーマット方式では、読みやすいプログラムを作ることを、常に心がけておくことが大切です。

### ■出力画面の改行

プログラムをコンパイルし、実行してください。次のように出力されます。

```
タノシイ C プ ロ グ ラ ミ ン グ
*EXIT (50)
```

実行結果を見ると ¥n が画面に表示されていません。¥nは改行を意味する特別な文字で、Cプログラムでは文字列中の¥nのある位置で改行が実行されます。BASICのPRINT文と違って、自動的な改行は行いません。

最初のprintf文を次のように変更してみましょう。

```
printf ( ' タノシイ ¥ n ' ) ;
```

画面は次のように2行に出力されます。

```
タノシイ
C プ ロ グ ラ ミ ン グ
*EXIT (50)
```

## 参 考

### ソースプログラムのセーブ（保存）

入力したCプログラム（ソースプログラム）は、本体のプログラムファイルエリアへ登録して保存します。

TEXTを押してテキストの機能選択画面にした後、(F)を押すとプログラムファイルのメニュー画面になります。

この画面で(S)を押すとセーブ（Save）が選ばれ、ファイル名の入力待ちになります。

ファイル名を入力して(ENTER)を押すと、プログラムがファイルされます。

なお、C言語のプログラムは、一般的にファイル名の拡張子を、C とします。

〈例〉 FILE NAME=TEST. C

このファイルをテキストエディタへ読み込むときは、Loadを使用します。

くわしくは、「第5章 TEXTモード」を参照してください。

### 3. 基本的なCプログラミング

この章では数値計算のプログラムを例に、Cプログラミングにおける変数の扱いかたを説明します。数値を出力する方法、変数宣言といった、Cプログラミングの基本的な機能を説明します。

#### 3. 1 整数を扱うプログラム

##### ■数値の出力

辺の長さが10の正方形の面積を求めるプログラム例です。

```
1 0 main ()
2 0 {
3 0 printf ("ヘン 10 ノ メンセキハ %d テ`ス¥n", 10*10);
4 0 }
```

プログラムを実行すると、画面には次のように出力されます。

```
ヘン 10 ノ メンセキハ 100 テ`ス
*EXIT (40)
```

printf 文の文字列の %d の部分が、100 に変わって出力されています。この100 という数値は10×10の演算結果です。%d は printf 文の中の文字列の後ろに続く値に置き換えられます。

```
printf ("ヘン 10 ノ メンセキハ %d テ`ス¥n", 10*10);
```

printf 文は出力する文字列の中に%記号があると、文字列と、で区切られたデータをその位置に出力します。%記号に続く文字 d は、データの表示形式を指示するための変換文字で、「データを10進整数で表示する」ことを意味します。

このように printf 文の文字列は、単なる文字列と違い画面への表示形式を示す働きをするもので、書式文字列と呼ばれます。

%d のような働きをするものを変換指定といいます。整数のための変換指定には次の種類があります。

| 変換指定 | 表示形式      |
|------|-----------|
| %d   | 10進数に変換する |
| %x   | 16進数に変換する |
| %o   | 8進数に変換する  |

##### ■変数を使う

変数を使って、上記のプログラムをもう少し一般的なプログラムにします。

```
1 0 main ()
2 0 {
3 0 int hen, menseki;
4 0 hen = 10;
5 0 menseki = hen * hen;
```

```
6 0 printf ("ヘン %d ノ メンセキハ %d テ`ス¥n", hen, menseki);
7 0 }
```

変数はいろいろなデータを記憶しておくための箱のようなものです。この箱には名前と、記憶しておくデータの種類（型）が決められます。これを変数宣言といいます。

行番号30で、変数の宣言をしています。BASICと違って、C言語ではプログラムで使う変数は、あらかじめ名前と変数の種類（データ型）を明確に宣言してからでないと使えません。変数宣言は次のように、データ型の指示に続いて変数名を書きます。

```
int hen, menseki;
↑      ↑      ↑
データ型 変数名
```

int は、変数 hen および menseki の扱うデータが整数値（小数点のつかない数値）であることを示すキーワードです。同じ型の変数を複数宣言する場合は各変数名を、で区切ります。宣言の終わりに ; が必要です。

行番号40は代入文です。= は代入を意味する演算子で、右辺の値が左辺の変数に格納されます。

```
10 ← 10
hen
```

行番号60の printf 文で変数 hen と menseki の内容を出力します。

```
ヘン 10 ノ メンセキハ 100 テ`ス
*EXIT (70)
```

printf 文では、変換指定の位置に変数の値が表示されます。複数の変数を出力する場合は、変数を、で区切ります。printf 文は記述された順に変数の値を表示します。

```
printf ("ヘン %d ノ メンセキハ %d テ`ス¥n", hen, menseki);
```

##### ■データの型と数値の範囲

C言語では小数点のつかない数値を整数型、小数点を含む数値を実数型（浮動小数点型ともいう）として、明確に区別します。これはデータ型によってコンピュータ内部で表現される形式が異なるためです。また整数型、実数型はそれぞれ表現する数値の大きさによって、さらに分類されます。代表的な型を次に紹介します。（数値の範囲はコンピュータによって異なります。）

| 分類  | データ型   | 扱う数値の範囲                                                | ビット長  |
|-----|--------|--------------------------------------------------------|-------|
| 整数型 | char   | -128～+127                                              | 8ビット  |
|     | int    | -32768～+32767                                          | 16ビット |
|     | short  | -32768～+32767                                          | 16ビット |
|     | long   | -2147483648～+2147483647                                | 32ビット |
| 実数型 | float  | $\pm 1 \times 10^{-99} \sim \pm 9.999 \times 10^{+99}$ | 32ビット |
|     | double | floatの仮数部有効桁が10桁                                       | 64ビット |

整数型の前に次の修飾子をつけることができます。

**unsigned** 正負の符号なし、つまり正の数のみ扱う

double 型に修飾子 long をつけることができますが、本機では double と同じ扱いになります。詳しくは「6. C言語の要点」を参照してください。

(注) 変数を使用する場合、各型で扱える数値の範囲を超えるような使いかたをすると正しい結果が得られません。

## ■変数の名前

変数の名前のつけかたに制限があります。

- 英文字で始まり、英字または数字で表す（カナ文字は使えない）
- 大文字と小文字は区別される
- 下線（`_`）は英字として扱う
- `+`、`-`、`&` など、特殊記号は使えない
- 名前の長さは31文字まで有効（32文字以降は無視）
- `int`、`double` といった、C言語のキーワードは使えない

### 正しい例

```
_max      : _ は英字に含む
int__var   : キーワードは名前の一部に使える
MyName     : 大文字と小文字が混在してもよい
number 2   : 先頭でなければ数字が使える
```

### エラーになる例

```
float      : キーワードは使えない
int-var    : - は特殊記号である
ヘンスウ a : カタカナは使えない
1 0 n      : 先頭に数字は使えない
```

本機で用意されているライブラリ関数名も名前に使えません。キーワード、ライブラリ関数については「6. C言語の要点」、「7. ライブラリ関数」を参照してください。

Cプログラミングでは、変数名は基本的に小文字で表します。また変数の名前はその働きがわかる名前をつけることが、わかりやすいプログラミングのコツです。

## ■キーボードからの数値入力

辺の長さをキーボードから入力するプログラムを示します。

```
1 0 main ()
2 0 {
3 0     int hen, menseki;
4 0     printf ( "ヘンノ ナカ サ?" );
5 0     scanf ( "%d", &hen );
6 0     menseki=hen*hen;
7 0     printf ( "ヘン %d ノ メンセキハ %d デ ス¥n", hen, menseki );
8 0 }
```

行番号5 0のscanf文が入力文で、キーボードからの入力を実行します。

```
scanf ( "%d", &hen );
```

この文が実行されると、プログラムはキーボードからの入力待ちの状態になります。キーボードから数値を入力すると、そのデータは変数henに格納されます。

%d はprintf文の場合と同様に、入力されたデータを10進整数として扱うことを指示する書式文字列です。

数値が格納される変数の名前に&記号がついていることに注目してください。

```
&hen
```

& はアドレス演算子と呼ばれるもので、&hen は変数henの記憶されている場所（メモリの番地）を表します。scanf文では入力されたデータを格納する「場所」を指示しなければなりません。

scanf文はBASICのINPUT文と違って、書式文字列の中に入力を促すメッセージを記述することはできません。そのため、行番号4 0のようにprintf文でメッセージを出力する必要があります。

## 3. 2 実数を扱うプログラム

### ■実数を使う

3教科のテストの平均点を求めるプログラムで、実数の使いかたを説明します。

```
1 0 main ()
2 0 {
3 0     int k=76, /* コクゴ */
4 0         s=88, /* スウガク */
5 0         e=81; /* エイゴ */
6 0     double ave;
7 0     ave= (k+s+e) / 3. 0;
8 0     printf ( "ヘイキンテン: %f ¥n", ave );
9 0 }
```

行番号3 0、4 0、5 0が各教科の点数を入れるための変数宣言です。ここでは変数の宣言と同時に代入を行っています。これを変数の初期化といいます。またそれぞれの行に `/*` と `*/` で囲まれたメッセージが記述されています。これをコメントといいます。コメントはプログラムの実行には何の影響も与えません。コメントはプログラムをわかりやすく記述するための働きをします。

行番号7 0で平均点を求めています。3教科の平均点ですから、合計を3で割ればよいのですが、わざわざ3. 0と小数点をつけて割っています。これは3で割った場合は整数どうしの演算ということになり、結果も整数、つまり小数点以下が切り捨てられてしまいます。これを防ぐために3. 0という実数型で割ることで、演算が実数型の演算となり、結果も小数点以下の値が求められます。

printf文の変換指定に %f が使われています。%f（および%e、%g）はdouble型のデータを出力するための変換指定です。%fは次のように表示形式を指定できます。

```
%6. 2f
```

これは正負の符号、小数点を含めた全体の桁数を6、小数点以下を2桁に指定することになります。実数型には%f以外にも %e（指数形式）、%gの変換指定があります。くわしくは「7. ライブラリ関数」のprintf関数を参照してください。



## ■指数を使う

科学計算、たとえば地球と太陽との距離のように大きな数値や、原子の重さのように非常に小さい数値を扱うには指数を使った計算が必要になります。

たとえば太陽からの光が地球に届く時間を求めてみると太陽と地球との距離、光の速度は次のように表します。

太陽と地球との距離  $1.496 \times 10^8$  [km] → 1. 4 9 6 e 8

光の速度  $2.998 \times 10^8$  [km/sec] → 2. 9 9 8 e 5

```
1 0 main ()
2 0 {
3 0     double d=1. 4 9 6 e 8 ;
4 0     double c=2. 9 9 8 e 5 ;
5 0     double t=d/c ;
6 0     printf ( 'time=%7. 2 f [sec] %n ', t ) ;
7 0 }
```

```
time= 499. 00 [sec]
*EXIT (70)
```

出力の変換指定を%7. 2 fとしたため、出力した数値の前が1桁空白になっています。桁数を指定すると、数値は右詰めで表示されます。

## 4. プログラムの流れを制御する

プログラムは上から順次実行されるとは限りません。プログラム中で与えられた条件によって実行される文が選択されたり、繰り返しが行われたりします。このようなプログラムの流れを制御する構文を説明します。

### 4. 1 条件文の使いかた

条件文は与えられた条件によって実行の流れを枝別れするための構文です。

#### ■if文

if文は次のようになります。

```
if (条件式) /* この条件が成立すれば */
    文1 ;    /* 文1を実行し          */
else        /* それ以外ならば        */
    文2 ;    /* 文2を実行する          */
```

実行される文は1つの文(単文という)でも、複数の文を含んでもかまいません。複数の文の場合は、それらの文を中カッコ、つまり { と } で囲みます。これを複文、またはブロックといいます。

else以下を省略できます。この場合、条件が不成立なら何もしないで条件文を抜けることになります。

分数では分母が0の場合は演算不能で、実行時エラーが発生します。次はこれを避けるためにif文を使ったプログラムです。

```
1 0 main ()
2 0 {
3 0     double a, b ;
4 0     printf ( 'a/b フォットメル %n' ) ;
5 0     printf ( 'a? ' ) ;
6 0     scanf ( ' %lf ', &a ) ;
7 0     printf ( ' %nb? ' ) ;
8 0     scanf ( ' %lf ', &b ) ;
9 0     if ( b == 0 )
10 0         printf ( " %n0 ティ フレマセン %n " ) ;
11 0     else
12 0         printf ( " %na / b = %f %n ", a/b ) ;
13 0 }
```

キーボードから変数a、bに数値を入力してください。行番号90の条件式b==0で、bの値が0であるかどうかを調べます。bの値が0以外なら割り算を実行し、0ならばエラーメッセージを出力してプログラムは終了します。

等しいかどうかをチェックする演算子は、等号 = を2つ続けます。C言語では「等しい」ことを意味する == と、「代入」を意味する = とは明確に区別します。

実行される文はキーワードifまたはelseより行頭を数桁下げます(本書では2桁)。これは実行のレベルが違うことを明らかにすることで、プログラムをわかりやすくするためのテクニックです。これを段付けといいます。

#### ■大小を比較する演算子

条件を与える式に、基本的には関係演算子を使います。関係演算子は2つの値を比較して、等しいか等しくないか、数値的にどちらが大きい小さいかを調べる演算子です。

| 式      | 真になる場合    |
|--------|-----------|
| a == b | aはbに等しい   |
| a != b | aはbに等しくない |
| a < b  | aはbより小さい  |
| a > b  | aはbより大きい  |
| a <= b | aはb以下である  |
| a >= b | aはb以上である  |

関係演算子は演算の結果として、真には数値1、偽には0を返します。

## ■ switch ~ case 文

条件による分岐先が多くなる場合は switch ~ case 文を使うとわかりやすくなります。switch ~ case 文は次のようになります。

```
switch (整数式)
{
    case 定数1: /* 整数式の値が定数1なら */
        文1; break; /* 文1を実行する */
    case 定数2: /* 整数式の値が定数2なら */
        文2; break; /* 文2を実行する */
    ...
    case 定数n: /* 整数式の値が定数nなら */
        文n; break; /* 文nを実行 */
    default: /* それ以外なら */
        文; /* 文を実行する */
}
```

break は無条件で制御構文を抜けるためのキーワードです。

次は暦の月を英語で表示するプログラムです。ただしスペースの関係で3月までの表示とします。

```
1 0 main ()
2 0 {
3 0     int month;
4 0     printf ( "ナンカ ツ ? " );
5 0     scanf ( "%d", &month );
6 0     switch (month) {
7 0         case 1:
8 0             printf ( "January ¥n" ); break;
9 0         case 2:
10 0             printf ( "February ¥n" ); break;
11 0         case 3:
12 0             printf ( "March ¥n" ); break;
13 0         default:
14 0             printf ( "ソノタ ¥n" );
15 0     }
16 0 }
```

## 4. 2 繰り返し文の使いかた

同じ作業を繰り返し実行するには、繰り返し文を使います。C言語の繰り返し文には for 文、while 文、do ~ while 文が用意されています。

### ■ while 文

while 文は与えられた条件式が成り立つ間、文を実行します。

while (条件式)

文;

if 文と同様、実行される文は単文でも、複文でもかまいません。

キーボードから数値を繰り返し入力し、その合計を求めるプログラムです。入力するデータの個数もキーボードから入力します。

```
1 0 main ()
2 0 {
3 0     int data, num;
4 0     int i = 0, sum = 0;
5 0     printf ( "データのコスウ? " );
6 0     scanf ( "%d", &num );
7 0     while ( i < num ) { ← 複文の始め
8 0         printf ( "¥nデータ? " );
9 0         scanf ( "%d", &data );
10 0        sum = sum + data;
11 0        i = i + 1;
12 0    } ← 複文の終わり
13 0    printf ( "¥nコウケイ %d ¥n", sum );
14 0 }
```

行番号100は「sum の内容と data の内容をたして、その結果で sum の内容を書き換える」ことを意味します。つまり入力データを順次、sum に合計していきます。行番号110も同様で、この文が実行されるたびに、変数 i の内容が1ずつ増加していきます。

行番号70の条件式では、i と num とを比較し i の値が num と等しくなった時点で繰り返しを終了し、行番号130に実行が移ります。

### ■代入演算子とインクリメント演算子

C言語では行番号100の文を次のように記述できます。

```
sum += data;
```

この記述法は、記述の簡素化だけでなく、プログラムの実行速度が速くなることが期待できます。このような演算子を代入演算子といいます。代入演算子は+演算子だけでなく-、/、\*、% (割り算の余りを求める) 演算子など、他の演算子にも適用できます。

また行番号110は次のように記述できます。

```
i++;
```

これをインクリメント演算子といい、この文が実行されるたびに i の内容は1ずつ増加します。インクリメント演算子はCプログラミングでは多用されます。

代入演算子、インクリメント演算子については「6. C言語の要点」を参照してください。

(注) 代入演算子を使用する場合で、右辺と左辺でデータ型が異なるときは、右辺の結果を左辺の型に変換した後で演算をするため、一般のC言語と結果が異なる場合があります。型の違うデータの演算は算術演算子による記述を行ってください。

## ■ for 文

for 文の一般的な書式は次のようになります。

```
for (文1 ; 条件式 ; 文2)
```

```
    本文 ;
```

while 文を次のように記述した場合と全く同じ働きをします。本文は単文でも、複文でもかまいません。

```
文1 ;                ← 繰り返しに入る前に実行される
```

```
while (条件式) {      ← 条件が成立している間、 { } を繰り返し実行する
```

```
    本文 ;
```

```
    文2 ;              ← 繰り返しを行うたびに実行される
```

```
}
```

文1で繰り返しを制御する変数(制御変数という)を宣言し、初期化します。条件式で制御変数のチェック、文2で制御変数のカウンタの働きをさせます。

```
10 main ()
20 {
30     int i;
40     for (i = 0 ; i < 10 ; i++) {
50         printf ('i=%d\n', i) ;
60     }
70 }
```

プログラムを実行すると、変数*i*の値が0から9まで変化していくことが確認できます。行番号40で繰り返しの制御が行われます。行番号40は「変数*i*の初期値を0とし、*i*が10より小さい間、*i*を1ずつ増加しながら繰り返す」という意味になります。

## ■ 制御文の中の制御文

条件文、繰り返し文はそれぞれ独立した1つの文ですから、条件文や繰り返し文の中で実行される文として使うことができます。これをネスティングといいます。

次はかけ算の九九を出力するプログラムです。for 文の中で for 文が実行されています。

```
10 main ()
20 {
30     int i, j;
40     for (i = 1 ; i < 10 ; i++) {
50         for (j = 1 ; j < 10 ; j++) {
60             printf ("%d * %d = %d\n", i, j, i * j) ;
70             printf (' ');
80         }
90     }
100 }
```

## 参考

上記のプログラムを実行すると、スピードが速すぎて表示の確認ができません。

このようなときは、for 文を使って表示時間を調整する方法があります。

たとえば、前記のプログラムに、次の行を追加します。

```
35     int k;
65     for (k = 0 ; k < 1000 ; k++) ;
```

65行は、「*k*が0から1000になるまで繰り返す」という意味です。

つまり、65行を繰り返し実行している間は、画面が変わりません。

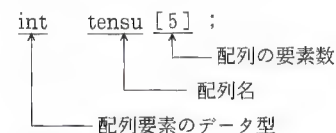
1000を他の値に変えれば、表示時間を変えることができます。

## 5. C言語らしいプログラミング

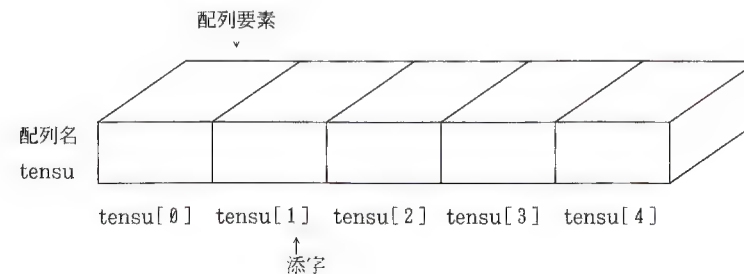
次に配列、関数などのCプログラミングでの特徴的な機能を中心に説明します。

### 5.1 配列

配列は同じ種類(データ型)のデータを大量に処理するとき、威力を発揮します。Cプログラミングでは変数宣言と同様に、配列の宣言が必要です。



これで配列名が *tensu*、配列要素のデータ型が *int*、配列要素の数が5である配列が宣言されたことになります。実際には配列のための領域がメモリに確保されます。



配列を宣言してから、その配列の各要素を指定するときは、配列名の後ろに大カッコ [ ] で囲んだ数値を書いて指定します。この数値を添字といいます。それぞれの配列要素は単独の変数と同じように扱えます。たとえば、配列 *tensu* の3番目の要素に代入するには次のようにします。

```
tensu [ 2 ] = 70 ;
```

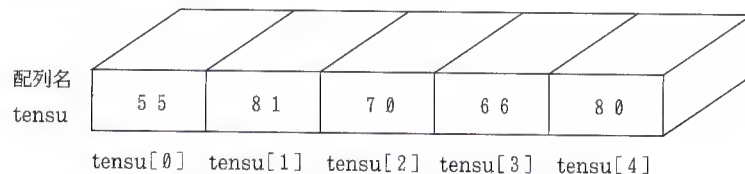
3番目の要素でありながら、添字は2になっていますがまちがいはありません。C言語では配列の要素は1ではなく0番目から始まるために、このような表記になるのです。

### ■配列の操作

次は、あるテストの得点での最高点を求めるプログラムです。

```
1 0   main ()
2 0   {
3 0     int   tensu [5] ;
4 0     int   i, n, max = 0 ;
5 0     tensu [0] = 5 5 ;
6 0     tensu [1] = 8 1 ;
7 0     tensu [2] = 7 0 ;
8 0     tensu [3] = 6 6 ;
9 0     tensu [4] = 8 0 ;
10 0    for   (i = 0 ; i < 5 ; i++) {
11 0        if   (max < tensu [i]) {
12 0            max = tensu [i] ;
13 0            n = i ;
14 0        }
15 0    }
16 0    printf ("サイコウハ  %dハ`ン : %dテン¥n", n, max) ;
17 0 }
```

各点数は次の図のように各要素に格納されます。



プログラムでは、行番号100から150において、配列要素を0番目から順にそれ以前の最大値 max と比較し、より大きい値で max の内容を書き換えていきます。for 文を抜けた後の max に、配列要素の中の最大値が残されます。

行番号30および50から90までの代入文は次のように記述できます。

```
int tensu [5] = {5 5, 8 1, 7 0, 6 6, 8 0} ;
```

このように配列宣言のときに各要素に一括して代入できます。これを配列の初期化といいます。

ただし、一般のC言語では、このように書けないものがあり、このように書くとエラーになる場合があります。

### ■文字の配列

Cプログラミングでは文字列は文字の配列として扱います。

```
1 0   main ()
2 0   {
3 0     char str [7] = "string" ;
4 0     int i ;
5 0     printf ("%s¥n", str) ;
6 0     for (i = 0 ; i < 6 ; i++)
7 0         printf ("%c-", str [i]) ;
8 0 }
```

行番号30で、文字の配列 str を宣言すると同時に、文字列の定数 **string** で初期化しています。Cプログラミングでは文字と文字の並びである文字列は区別しなければなりません。文字定数の場合はシングルクォーテーションマーク ' ' で1文字を囲みます。文字列定数はダブルクォーテーションマーク " " で文字の並びを囲みます。

char c = 'A' ; : 文字定数

char str [7] = "string" ; : 文字列定数

行番号50で文字列 str の内容を出力します。%s が文字列のための変換指定です。

行番号60、70で文字列の要素を順次出力します。%c が1文字を出力するための変換指定です。

なお for 文は繰り返し実行する文が単文のため { } は省略しています。

### ■文字コード

文字の配列がどのようなになっているか、もう少し詳しく調べましょう。

```
1 0   main ()
2 0   {
3 0     char str [3] = "AB" ;
4 0     int i ;
5 0     for (i = 0 ; i < 3 ; i++)
6 0         printf ("%c : %x¥n", str [i], str [i]) ;
7 0 }
```

画面には次のように表示されます。

```
A : 41
B : 42
: 0
*EXIT (70)
```

文字にはそれぞれ文字コードが決められており、文字配列の各要素には文字コードを意味する数値が格納されます。画面の1行目、2行目は文字配列の要素 'A'、'B' とその文字コードが16進数で表示されます。なお、文字コードについては「キャラクタ・コード表」を参照してください。

画面の3行目に注目してください。配列の3番目の要素には数値0が格納されています。0はC言語では文字列の終わりを表すことを意味する大事な働きをします。0を特殊文字としてヌル文字と呼び、'¥0' と記述することがあります。したがって文字配列の要素数は、(格納する文字数 + 1) 以上でなければなりません。



## 5.2 関数

関数はCプログラミングの中で、最も基本的な実行単位です。関数は1つのまとまった仕事をしてくれる、独立した副プログラムです。作成された関数には名前がつけられ、新しい命令としてプログラムの中で使うことができます。

関数はユーザー（利用者）が作成する以外に、C言語のシステムによって提供されるライブラリ関数があります。ライブラリ関数は一般的に役に立つ関数を集めたもので、これまで使ってきた `printf()`、`scanf()` もその1つです。本機で用意してあるライブラリ関数については、「7. ライブラリ関数」を参照してください。

関数は互いに対等ですが、`main()` だけは特別な関数でプログラムの実行はこの関数から始まります。そのため主（メイン）関数と呼びます。Cプログラムは `main()` を中心に、ユーザー関数、ライブラリ関数を組み合わせて作っていきます。

### ■関数を作る

関数は一般に、呼び出し側からデータを受け取ってなんらかの処理をした後、結果を返します。2つの整数の和を求める関数 `plus()` を例に、関数の作りかた（定義）と使いかたを説明します。

```
int plus(int a, int b)
{
    return (a + b);
}
```

先頭行が関数の定義にとって重要な意味を持ちます。

```
int  plus(int a, int b)
↑      ↑      ↑      ↑
関数の型 関数の名前 引数の並び（引数がない場合は空欄のまま）
```

上の例では「関数の名前は `plus` で、2つの `int` 型の値を受け取り、処理した結果を `int` 型の値として返す」ことを表しています。なお、値を受け取る変数を引数（ひきすう）といい、関数の中では他の変数と全く同じ働きをします。関数の型は関数が返す値の型を意味します。`plus()` 関数は実行結果を `int` 型データとして返します。

関数の内部での処理は `{ }` で囲んだブロックの中に記述します。

```
{
    return (a + b);
}
```

`return` 文は関数の実行を終わる働きをします。そのとき `return` のすぐ後ろの値を関数の値（戻り値）として返します。`plus()` は `(a + b)` の値を返して実行を終わります。

なお、関数名についての規則は変数名のつけかたと同じです。

### ■関数の呼び出し

関数 `plus()` を使ったプログラムの全体を示します。

```
1 0  main()
2 0  {
```

```
3 0  int plus(int, int);
4 0  int x;
5 0  x = plus(10, 20);
6 0  printf("10+20=%d\n", x);
7 0  }
8 0  /* カンスウ テイキ */
9 0  int plus(int a, int b)
10 0 {
11 0     return (a + b);
12 0 }
```

関数を呼び出す側では、あらかじめ呼び出す関数を宣言しておかなければなりません。行番号30が関数の宣言で関数定義の先頭行と同じ形式で宣言します。ただし、関数宣言では引数は型名だけでもかまいません。また宣言の最後にセミコロン `;` が必要です。

行番号50で関数の呼び出しが行われ、2つの数値が渡されます。

```
呼び出し側  x = plus(10, 20);
              ↑      ↓
              戻り値  引数
関数側      plus(int a, int b);
```

関数の戻り値は変数 `x` に代入されます。関数自身が、変数のように値を持っている様子に注目してください。

呼び出す側から関数へ渡す値を実引数、関数側の引数を仮引数と呼びます。実引数は変数でもかまいません。お互いの引数はそれぞれ呼び出す側と呼び出される側の窓口です。関数内部で仮引数の値が変化しても実引数に影響を与えません。

### ■戻り値のない関数

次の `message()` のように戻り値を持たない関数を `void` 型関数といい、関数名の前にキーワード `void` を記述します。

```
1 0  main()
2 0  {
3 0      void message(void);
4 0      message();
5 0  }
6 0  /* void カンスウ */
7 0  void message(void)
8 0  {
9 0      printf("void カンスウ");
10 0 }
```

`message()` は引数を持たない関数でもあります。この場合は、引数がないことをはっきりさせるため、関数の定義および宣言の引数並びには `void` と記述します。ただし、関数呼び出し（行番号40）では引数並びは空欄にします。

## ■変数の有効範囲

関数の中で宣言された変数の有効範囲（代入したり読み出したりできる範囲）は関数内部に限られます。

このような変数を局所（ローカル）変数といいます。

```

1 0  main ()
2 0  {
3 0      void fun (void) ;
4 0      int i ;
5 0      for (i = 0 ; i < 3 ; i++) {
6 0          printf ( 'main i=%d ', i) ;
7 0          fun () ;
8 0      }
9 0  }
100 /* カンスウ */
110 void fun (void)
120 {
130     int i = 10 ;
140     printf ( " fun i=%d \n ", i) ;
150 }
```

プログラムを実行すると画面には次のように出力されます。

```

main i=0 fun i=10
main i=1 fun i=10
main i=2 fun i=10
*EXIT (90)
```

このプログラムでは main () 関数と fun () 関数とで同じ名前の変数 i を使っていますが、それぞれの変数は互いに独立しています。main () 関数で i の値が変化しても、呼び出された fun () 関数の i は影響を受けていません。

局所変数は、変数宣言された関数の外から操作できません。また関数間で同じ名前を使っても、お互いが影響し合うことはありません。したがって関数ごとに独立してプログラミングできるため、プログラミング効率がよくなり、またプログラムのミス（バグ）も発生しにくくなります。

関数の外部で宣言される変数を大域（グローバル）変数といい、どの関数からでも共通して参照できます。

```

1 0  int g = 0 ;
2 0  main ()
3 0  {
4 0      void fun (void) ;
5 0      int i ;
6 0      for (i = 0 ; i < 3 ; i++) {
7 0          printf ( 'g=%d \n ', g) ;
8 0          fun () ;
9 0      }
```

```

1 0 0 }
1 1 0 /* カンスウ */
1 2 0 void fun (void)
1 3 0 {
1 4 0     g++ ;
1 5 0 }
```

プログラムを実行すると画面には次のように出力されます。

```

g=0
g=1
g=2
*EXIT (100)
```

変数 g は fun () 関数で変更することができ、main () から fun () が呼び出されるたびに g の値が1ずつ増加していきます。

大域変数は関数の間で共通の変数が必要な場合など、特別な場合にしか使いません。大域変数の多用は、プログラムをわかりにくくし、関数の独立性を損ないます。

Cプログラミングは必要な関数を作り、それを組み合わせて全体のプログラムを構成するというのが基本です。そのため使う変数はできるだけ局所的にすることで、それぞれの関数の独立性を保つことを心がけなければなりません。

## ■変数の存在期間

これまで使ってきた局所変数は、宣言した関数が呼び出されるたびに生成され、その実行が終わるたびに消滅します。このような変数を自動（auto）変数とも呼びます。

変数宣言で、キーワード **static** を使うとその変数は静的変数になります。

```
static int s ;
```

静的変数はプログラムの終了時まで存在します。関数内で宣言された静的変数は、その関数の実行が終了した後も値を保持し続け、次に同じ関数が呼び出されるとその値が受け継がれます。

```

1 0  main ()
2 0  {
3 0      void count (void) ;
4 0      int i ;
5 0      for (i = 0 ; i < 3 ; i++)
6 0          count () ;
7 0  }
8 0 /* カンスウ テイキ */
9 0 void count (void)
100 {
110     int a = 0 ;
120     static int s = 0 ;
130     printf ( " a=%d s=%d \n ", a, s) ;
140     a++ ; s++ ;
150 }
```

画面には次のように表示されます。

```
a=0 s=0
a=0 s=1
a=0 s=2
*EXIT (70)
```

自動変数 `a` の値は関数呼び出しのたびに 0 に初期化されますが、静的変数 `s` の値は前の内容に 1 ずつ加算された値になっています。

### 5.3 ポインタ

ポインタはポイントするもの、すなわち何かを指し示すものという意味です。プログラミングでは、データが記憶されている場所を指し示すことを意味します。最初はデータが記憶されているメモリのアドレス(番地)のことでありと理解しておけばよいでしょう。

ポインタ(アドレス)を格納するためには特別な変数が必要となります。それをポインタ変数といいます。Cプログラミングの説明では、ポインタとポインタ変数はときどき区別されないで用いられることがありますので注意してください。

#### ■変数のアドレスとポインタ

変数の宣言によって、変数にはメモリのある場所が割り当てられます。変数への代入とは、実際にはその場所にデータを記憶することにほかなりません。変数に割り当てられた記憶場所のアドレスはアドレス演算子 `&` を使うことで得られます。たとえば変数 `a` のアドレスは次の式から得られます。

```
&a
```

ポインタ変数は他の変数と同じく、変数宣言しなければなりません。このとき、どのデータ型変数でアドレスを格納するかも明示する必要があります。たとえば `int` 型の変数を扱うポインタ変数は次のように宣言します。

```
int *p;
```

`*` はポインタ演算子と呼ばれ、`p` がポインタ変数であることを示します。また `*` 演算子はポインタの前につけることで、そのポインタの指し示すアドレスに格納されている値を返します。

```
10 main()
20 {
30 int a=5;
40 int *p;
50 p=&a;
60 printf("&a=%p: a=%d\n", &a, a);
70 printf("p=%p: *p=%d\n", p, *p);
80 }
```

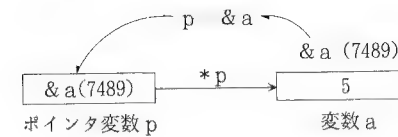
プログラムが実行されると画面は次のようになります。

(注) 7489 (アドレス) はメモリが使用

されている状態によって変わります。

```
&a=7489: a=5
p=7489: *p=5
*EXIT (80)
```

行番号 60、70 の `printf` 文中の `%p` はポインタを出力するための変換指定です。出力画面から変数 `a` のアドレスが16進数で 7489 に割り当てられていることがわかります。また `*p` がポインタの指し示す内容、すなわち変数 `a` の値を表していることも理解できます。



#### ■配列とポインタ

ポインタは配列と密接な関係があります。ポインタを使って配列操作を行うと効率のよい、実行速度の速いプログラムを作ることができます。

配列要素はメモリ上の連続した領域に格納されます。またC言語では配列名は配列の先頭要素のアドレスを表します。文字配列 `str[]` を例にとると、`str` と先頭要素のアドレス `&str[0]` とは同じ意味になります。

ある配列要素を参照するには、先頭要素の添字を 0 としたときの要素の位置を添字として指示します。

このような配列操作はポインタを使っても全く同じように行うことができます。

```
10 main()
20 {
30 int i;
40 char str[7] = "string";
50 char *ptr;
60 ptr = str;
70 for(i=0; i<7; i++)
80 printf("%c", *(ptr+i));
90 }
```

行番号 60 で配列の先頭アドレスをポインタ変数 `ptr` に代入しています。行番号 80 では順次ポインタに加える値を増加させながら、その指し示す内容を出力させています。

配列要素とポインタとの関係を次の図に示します。配列の添字と、ポインタに加える値(オフセット値)が同じであることに注目してください。

|     |        |          |          |       |          |
|-----|--------|----------|----------|-------|----------|
| str | 's'    | 't'      | 'r'      | ...   | '\0'     |
|     | str[0] | str[1]   | str[2]   | ..... | str[6]   |
|     | *ptr   | *(ptr+1) | *(ptr+2) | ..... | *(ptr+6) |

インクリメント演算子を使うと、ポインタによる配列操作がスマートに行えます。

```
ptr++
```

たとえば、上記の80行を次のようにします。

```
80 printf("%c", *ptr++);
```

次のプログラムは与えられた文字列の長さを求めるもので、ポインタによる配列操作の典型的なスタイルといえます。

```
10 main()
20 {
30 char str[10] = "pointer";
```

```

4 0    char  *ptr;
5 0    int   n = 0;
6 0    ptr = str;
7 0    while (*ptr != 0) {
8 0        ptr++;
9 0        n++;
10 0    }
11 0    printf("モシ* レッノ ナカ* サハ %d %n", n);
12 0 }

```

行番号7 0の繰り返しの条件式は、ポインタの指し示す内容が0、すなわち、文字列の終わりであるかどうかを調べます。

行番号7 0から1 0 0までのリストは次のように簡潔に表示できます。

```

while (*ptr++ != 0)
    n++;

```

条件式は\*ptrの値を調べた後、ポインタ変数ptrの値を1つ増加します。つまり、条件のチェックと行番号8 0のポインタ操作を同時に行っています。

## 6. C言語の要点

### 6. 1 本機のC言語仕様

本機のC言語の仕様は標準的なC言語に合わせるようにしていますが、次の項目は例外となります。

- ① 型定義 (typedef) 宣言できない。
- ② 列挙型 (enum) 宣言できない。
- ③ volatile 指定できない。
- ④ ビットフィールド宣言できない。
- ⑤ 配列型の auto 変数を初期化できる。(その場合は配列サイズを明示しなければならない。)
- ⑥ 構造 (struct) 型/共用 (union) 型は、引数や関数値にできない。
- ⑦ 構造 (struct) 型/共用 (union) 型は、条件演算子 a ? b : c で使用できない。
- ⑧ ブロック ( { } ) 内の宣言は、トップレベルのブロックでしかできない。
- ⑨ プリプロセッサの機能は、引数なしの # define、# include、# if ~ # elif ~ # endif、# ifdef ~ # elif ~ # endif である。
- ⑩ ライブラリ名・ストリーム名と同名の関数名・変数名は宣言できない。
- ⑪ main() は引数を持たない。
- ⑫ 定数式では次の演算子のみ有効である。

```

整数演算    単項 -    単項 +    ~
              *    /    %    +    -    <<    >>    &    ^    !

実数演算    単項 -    単項 +

```

(注) 定数式では、整数と実数間の型変換はできません。

- ⑬ C領域以外のメモリへの書き込みは禁止されている。
- ⑭ ポインタでの関数呼び出しで、C領域以外のポインタ値は許されていない。
- ⑮ 秒記号 ( " ) は扱えない。
- ⑯ 以下の制限がある。

# define 文において

|              |       |
|--------------|-------|
| 個数           | 3 2   |
| 名前の文字数       | 3 1   |
| 本体の文字数       | 3 1   |
| 名前の本体の保存バッファ | 2 5 5 |
| 展開バッファ       | 8 0   |

# include 文において

|        |   |
|--------|---|
| 入れ子の深さ | 2 |
|--------|---|

# if、# ifdef 文において

|        |   |
|--------|---|
| 入れ子の深さ | 2 |
|--------|---|

|                                   |     |
|-----------------------------------|-----|
| 定数の数字の数                           | 7 0 |
| 名前の文字数                            | 3 1 |
| 文字列の文字数                           | 8 0 |
| 型修飾 (*, [ ], ( ))                 | 8   |
| 1 関数内の goto ラベル                   | 8   |
| 最外側の switch 文内の case ラベル          | 3 2 |
| 構造体、共用体、配列、関数での宣言の深さ              | 4   |
| 構造体、配列での初期化変数の深さ                  | 8   |
| プロトタイプの深さ                         | 3   |
| 最外側 ( ) 内のプロトタイプ内の宣言数             | 3 2 |
| 次の構文での入れ子の深さ                      |     |
| (if, while, do, for, switch, { }) | 1 5 |
| 式の演算子の深さ                          | 1 6 |
| 式の演算子のオペランドの深さ                    | 3 2 |
| 式の深さ (tree 形式で)                   | 3 2 |

### 6. 2 キーワード

次の識別子 (名前) はC言語のキーワード (予約語) として使われるので、別の用途 (変数名、関数名など) に使えません。

|          |        |          |          |
|----------|--------|----------|----------|
| auto     | double | int      | struct   |
| break    | else   | long     | switch   |
| case     | enum   | register | typedef  |
| char     | extern | return   | union    |
| const    | float  | short    | unsigned |
| continue | for    | signed   | void     |
| default  | goto   | sizeof   | volatile |
| do       | if     | static   | while    |



また、次のキーワードも本機では使用できません。

enum          typedef          volatile

### 6.3 定数

定数には整数定数、文字定数、文字列定数、実数定数、列挙定数（本機では使えない）があります。

| 表 記 法 | 表 記 例                | 備 考                                |
|-------|----------------------|------------------------------------|
| 整数    | 1 0 進                | 1 2 8                              |
|       | 1 6 進                | 0 x 2 f                            |
|       | 8 進                  | 0 5 6                              |
| 実 数   | 1. 4 9 6, 2. 3 4 e 7 | <b>e</b> または <b>E</b> をつけて指数表記もできる |
| 文 字   | 'A', '¥n'            | シングルクォーテーションで1文字を囲む                |
| 文字列   | 'Programming'        | ダブルクォーテーションで文字の並びを囲む               |

整数定数では数値の後ろに **L** または **L** をつけると、long（32ビット）として扱われます。それ以外は大きさに合わせてintまたはlongです。

〈例〉 2 6 3 4 8 L

実数定数では数値の後ろに **f** または **F** をつけると、float（32ビット）として扱われます。それ以外ではdoubleです。

〈例〉 2. 6 5 f

文字定数はint（16ビット）で表現され、ASCIIコードが使用されています。

C言語では次の表のエスケープ列を使った制御文字を出力することができます。エスケープ列は文字定数として扱われます。

| 制御文字 | 値(16進)  | 機 能                  |
|------|---------|----------------------|
| ¥b   | 0 x 0 8 | 復帰（カーソルを1文字前へ移動）     |
| ¥n   | 0 x 0 A | 復帰改行（カーソルを次の行の先頭へ移動） |
| ¥r   | 0 x 0 D | 復帰（カーソルをその行の先頭へ移動）   |
| ¥t   | 0 x 0 9 | 水平タブ（カーソルを次の水平タブへ移動） |
| ¥¥   | 0 x 5 C | 文字¥を出力               |
| ¥'   | 0 x 2 C | 文字'を出力               |
| ¥'   | 0 x 2 2 | 文字'を出力               |
| ¥?   | 0 x 3 F | 文字?を出力               |
| ¥ddd |         | 8進数ddd（3桁）に対応する文字を出力 |
| ¥xhh |         | 16進数hhに対応する文字を出力     |

### 6.4 扱える数値の範囲

整数型で扱える数値の範囲は次のとおりです。

| データ型  | 扱う数値の範囲                   | 符号なし (unsigned) | 扱う数値の範囲        |
|-------|---------------------------|-----------------|----------------|
| char  | -128 ~ +127               | unsigned char   | 0 ~ 255        |
| short | -32768 ~ +32767           | unsigned short  | 0 ~ 65535      |
| int   | -32768 ~ +32767           | unsigned int    | 0 ~ 65535      |
| long  | -2147483648 ~ +2147483647 | unsigned long   | 0 ~ 4294967295 |

実数型で扱える数値の範囲は次のとおりです。

| データ型        | 扱う数値の範囲                            |
|-------------|------------------------------------|
| float       | ± 1 e - 99 ~ ± 9. 999 e + 99       |
| double      | ± 1 e - 99 ~ ± 9. 999999999 e + 99 |
| long double | ± 1 e - 99 ~ ± 9. 999999999 e + 99 |

### 6.5 式と文

Cプログラミングでは式と文の違いを理解しておくことが大切です。式は演算の結果としての値を持ちます。式の最後にセミコロン ; をつけたものが文で、実行の基本単位となります。式が値を持つため、次のような文が許されます。

a = x = 5 ;

この文ではまず右の代入式 (x = 5) が評価された後、式の値である5がaに代入されます。

### 6.6 演算子

C言語には豊富な演算子が用意されています。演算子の意味と働きを説明します。

#### ①算術演算子

| 演算子 | 意味 | 使用例   | 意 味          |
|-----|----|-------|--------------|
| +   | 加算 | a + b | a に b を加える   |
| -   | 減算 | a - b | a から b を引く   |
| *   | 乗算 | a * b | a に b を掛ける   |
| /   | 除算 | a / b | a を b で割る    |
| %   | 剰余 | a % b | a を b で割った余り |

#### ②代入演算子

| 演算子 | 使用例    | 意 味                  | 算術演算子による表記 |
|-----|--------|----------------------|------------|
| =   | a = b  | a に b を代入            | —          |
| +=  | a += b | a に b を加えた結果を a に代入  | a = a + b  |
| -=  | a -= b | a から b を引いた結果を a に代入 | a = a - b  |
| *=  | a *= b | a に b を掛けた結果を a に代入  | a = a * b  |
| /=  | a /= b | a を b で割った結果を a に代入  | a = a / b  |
| %=  | a %= b | a を b で割った余りを a に代入  | a = a % b  |



## ⑨いろいろな演算子

### ■アドレス演算子とポインタ演算子

C言語にはアドレスを直接操作する演算子があります。

& : 変数および配列要素のアドレスを取り出す (アドレス演算子)

\* : ポインタが指し示す変数への参照 (ポインタ演算子)

### ■条件演算子

条件演算子は if 文と同じような働きをする演算子です。

a ? b : c

条件式 a が真なら式 b を、偽なら式 c を評価した値を返します。

a = x >= 0 ? 1 : 0

x が 0 または正ならば 1 が、負であれば 0 が a に代入されます。

### ■sizeof 演算子

オブジェクトを格納するのに必要なバイト数を求める演算子です。

sizeof (型) : データ型の大きさをバイト数で返す

sizeof 式 : 式の値を格納するのに必要なバイト数を返す

たとえば次の例では s に 2 (バイト) が代入されます。

s = sizeof (int) ;

## ⑩演算子の優先順位と結合規則

Cプログラミングでは演算子は数や機能が豊富なため、使いかたに慣れるのは大変です。特に演算子を組み合わせた式では、演算子の優先度に注意を払わなければなりません。

表に演算子の優先順位と結合規則を示します。

| 優先度 | 演算子の種類  | 結合規則 | 演 算 子                        |
|-----|---------|------|------------------------------|
| 高い  | 1 次式演算子 | 左から右 | ( ) [ ] ->                   |
|     | 単項演算子   | 右から左 | ! ~ ++ -- + - (型) * & sizeof |
|     | 2 項演算子  | 左から右 | * / %                        |
|     |         | 左から右 | + -                          |
|     |         | 左から右 | << >>                        |
|     |         | 左から右 | < <= > >=                    |
|     |         | 左から右 | == !=                        |
|     |         | 左から右 | &                            |
|     |         | 左から右 | ^                            |
|     |         | 左から右 | &&                           |
|     |         | 左から右 | ,                            |
|     | 条件演算子   | 右から左 | ? :                          |
| 低い  | 代入演算子   | 右から左 | = += -= その他                  |
|     | コンマ演算子  | 左から右 | ,                            |

表において、同じ行の演算子の優先度は同じです。結合の方向とは同じ優先度の演算子が続いたときに、式の評価の方向をいいます。次の式は右の式のように評価されます。

a + b - c ----> (a + b) - c : 左から右へ結合

a = b += c ----> a = (b += c) : 右から左へ結合

## 6. 7 いろいろな構文

Cプログラムでの実行の流れは次の制御構文に分類できます。

逐次構文 : 並べられた順に、文を実行する。

選択構文 : 条件によって、実行の流れを分岐する。(条件文)

ループ構文 : ある一定の条件下で、実行を繰り返す。(繰り返し文)

### ①単文と複文

ほとんどの文は式の文で、次のように文の最後はセミコロン ; で終わります。

式 ;

複数の文を 1 つの文として扱えるようにするために複文が用意されています。複文は次のように単文をカッコ { } で囲みます。複文はブロックともいいます。

```
{
  文
  . . . .
  文
}
```

一般にはブロック内での変数宣言ができますが、本機ではトップレベルのブロックでしか許されていません。

### ②選択文

C言語では if ~ else 文、switch ~ case 文の選択文が用意されています。

#### ■if ~ else 文

if ~ else 文には次のような形式があります。

1. if (式) 式が真なら文を実行し、偽なら単に通り返ける。

文

2. if (式) 式が真なら文 1 を、偽なら文 2 を実行する。

文 1

else

文 2

3. if (式 1) 式 1 が真なら文 1 を、そうでなく式 2 が真なら文 2 を、式 1、式 2 とも偽なら文 3 を実行する。

文 1

else if (式 2)

文 2

else

文 3

## ■ switch ～ case 文

switch ～ case 文は次の形式になっています。

```
switch (条件式) {
    case 定数式 1 : 文 1
        break ;
    case 定数式 2 : 文 2
        break ;
    . . . . .
    case 定数式 n : 文 n
        break ;
    default : 文
}
```

switch 文は最初に条件式を評価し、その値と一致する定数式（ラベル）の文を実行します。

実行した後、break 文で switch 文を抜けます。break 文がないとその後に続くラベルの文へ進みます。

条件式に一致する定数式がない場合はキーワード default に続く文を実行します。

switch ～ case 文の条件式、定数式の型は整数型でなければなりません。

また、評価は unsigned int 型に変換して行われます。

## ③ 繰り返し文

C 言語では for 文、while 文、do ～ while 文が用意されています。

### ■ for 文

for 文は次の形式になります。

```
for (式 1 ; 式 2 ; 式 3)
    文
```

for 文は最初に 1 回だけ式 1 を評価します。式 2 が真であれば文を実行した後、式 3 を実行します。同様の動作を式 2 が偽になるまで繰り返します。

### ■ while 文

while 文は次の形式になります。

```
while (条件式)
    文
```

条件式を評価し、真である間、文を繰り返し実行します。

### ■ do ～ while 文

do ～ while 文は次の形式になります。

```
do
    文
while (条件式) ;
```

文を実行した後、条件式を評価します。条件式が偽になるまで文を繰り返し実行します。

## ④ ジャンプ文

ジャンプ文はプログラムの実行の流れを無条件に移す働きをします。ジャンプ文として goto 文、continue 文、return 文があります。

### ■ goto 文

goto 文は次のように使います。

```
goto ラベル ;

. . . . .

ラベル : 文
```

実行の流れが無条件にラベルの位置にジャンプします。ジャンプする先は同じ関数の中に限定されます。

### ■ continue 文

continue 文は繰り返し文の中のみ置くことができます。continue 文は繰り返し文で、ムダな実行を省略して実行速度を上げるためなどに使います。

```
. . . . .
for (i = 0 ; i < 100 ; i++) {
    . . . . .
    if (i % 2 == 0)
        continue ;
    printf ("%d %n", i) ;
}
. . . . .
```

上の例では i が奇数のときだけ数値が出力されることになります。

### ■ break 文

繰り返し文や switch ～ case 文で、構文を抜け出し 1 つ外側の実行単位に移るために使います。

```
. . . . .
for (i = 0 ; i < 100 ; i++) {
    if (a[i] < 0)
        break ;
    . . . . .
}
. . . . .
```

上の例では配列の要素が負になったら for 文を抜けます。

### ■ return 文

return 文は実行中の関数を抜け、その関数を呼び出した関数に実行を戻します。次のように戻り値がある場合は return 文の後ろに記述します。

```
return (式) ;
```

( ) は省略できますが、つけておいたほうがわかりやすくなります。戻り値がない場合に ( ) をつけるとエラーになります。



## 6. 8 記憶クラス

C言語では、変数をメモリのどの位置に配置するかによって、自動的と静的の2つの記憶クラスに分かれます。またプログラムで宣言された変数の場所が、関数またはブロックの中か外かによって、有効範囲が決められます。

| 種 類      | 有 効 範 囲                                            | 存 在 時 間                                                           |
|----------|----------------------------------------------------|-------------------------------------------------------------------|
| auto     | 宣言された実行単位の中で有効。<br>ただし、本機ではトップレベル以外のブロックでの宣言はできない。 | 実行単位から抜けた後は消滅する。                                                  |
| register | auto 変数と同じであるが、実行速度を上げるために用いる。ただし本機では auto とみなされる。 | auto と同じ。                                                         |
| static   | 宣言された実行単位の中で有効。<br>ただし、本機ではトップレベル以外のブロックでの宣言はできない。 | プログラム実行中は消滅せず、値を保存する。                                             |
| extern   | 宣言された実行単位で有効。<br>ただし、本機では「記憶クラス省略」とみなされる。          | プログラム実行中は消滅せず、値を保存する。<br>ただし、本機では関数内で宣言すると auto、関数外では static と同じ。 |

## 6. 9 多次元配列

配列の配列を2次元配列、またその配列を3次元配列といいます。C言語ではこのように多次元配列が使えます。本機では8次元までの配列が使えます。

2次元配列の宣言は次の例のように、配列の大きさを [ ] で囲んで並べます。

```
char color[3][6]
      ↑   ↑
      行   列
```

1次元配列と同様、宣言のときに初期化ができます。

```
char color[3][6] = { 'white', 'red', 'blue' } ;
```

図のように配列要素にデータが格納されます。

|             |     |             |             |             |             |             |             |
|-------------|-----|-------------|-------------|-------------|-------------|-------------|-------------|
|             |     |             | color[0][1] |             | color[0][3] |             | color[0][5] |
|             |     | color[0][0] | ↓           | color[0][2] | ↓           | color[0][4] | ↓           |
| color[0][i] | 'w' | 'h'         | 'i'         | 't'         | 'e'         | '\0'        |             |
| color[1][i] | 'r' | 'e'         | 'd'         | '\0'        | '\0'        | '\0'        |             |
| color[2][i] | 'b' | 'l'         | 'u'         | 'e'         | '\0'        | '\0'        |             |

## 6. 10 構造体

配列は同じ型のデータを扱うのに便利なデータ型であるのに対して、構造体、共用体は性質の異なるデータの集まりを、ひとつのデータ構造として表現するためのデータ型です。

たとえば野球選手の打撃成績を表現するデータ型は次のようになります。

```
struct batting {
    char *name;
    float ave;
    int homer;
};
```

**struct** は構造体であることを示すキーワードです。**batting** は構造体のタグといい、定義される構造体の形式につける名前です。構造体の要素は { } の中に記述します。選手の名前を \*name、打率を ave、ホームラン数を homer の各要素に格納します。

構造体型の変数は次のように宣言します。

```
struct batting batter;
```

タグ名を省略して直接、構造体変数を宣言できます。

```
struct {
    char *name;
    float ave;
    int homer;
} batter;
```

各要素への参照は次のように、変数名と要素名を . でつないで表記します。

変数名. 要素名

選手名、打率、ホームラン数を代入してみましょう。

```
batter.name = "ナカハラ";
batter.ave = 3.14;
batter.homer = 40;
```

## 6. 11 プリプロセッサ

C言語の処理系ではソースプログラムの中にプロセッサ指令があると、プログラムをコンパイルする前に指令に従った「前処理」を行います。いくつかのプリプロセッサのうち、「文字列定義／マクロ定義」を行う #define、「ファイルの取り込み」を行う #include と「条件コンパイル」を行う #if、#ifdef、#ifndef を説明します。

### ■ #define 指令

#define は指定された文字列の置き換えを行います。

```
#define 文字列1 文字列2
```

文字列2は文字列1に置き換えられてからコンパイルされます。#define は次の例のように、プログラム中で使われる定数を文字列に置き換える場合などによく使われます。

```
#define PI 3.141592
```

文字列 P I はプログラムの中では円周率を意味する定数として使うことができます。

#### 置き換え

```
s = P I * r * r;      →      s = 3. 1 4 1 5 9 2 * r * r;
```

# define 指令では次のことに注意が必要です。

- 文字列 1 は大文字で書くことが習慣となっている。
- 構文の途中で改行はできない。

本機ではあらかじめ、次のように各文字列が定義されています。

```
# define N U L L      0
# define E O F        - 1
# define F I L E      int
```

また本機では次のような引数付きマクロを定義できません。

```
# define S Q R ( x )  (( x ) * ( x ))
```

### ■ # include 指令

# include 指令は指定したファイルを、ソースプログラムのその位置に取り込みます。一般のC言語処理系では、ライブラリ関数を使うためのヘッダファイルを読み込むためによく使われますが、本機ではヘッダファイルの読み込みは不要です。

ファイルの読み込みは次のようにします。

```
# include " ファイル名 "
```

### ■ # if ~ # elif ~ # else ~ # endif 指令

条件文により、コンパイルするテキストを指定します。

```
# if 条件式
    テキスト
[ # elif 条件式
    テキスト ]
[ # else
    テキスト ]
# endif
```

条件式を満たすところに記述されているテキストのみをコンパイルします。なお、条件式を満たすところがあれば、次の # elif の条件式を判別します。# elif、# else は省略できます。

### ■ # ifdef、# ifndef 指令

```
# ifdef 文字列
    テキスト
[ # else
    テキスト ]
# endif
```

文字列が # define 指令で指定されているかどうかにより、テキストをコンパイルするかないかを指定します。

# ifdef は文字列が指定されているときにコンパイルを行い、# ifndef は文字列が指定されていないときにコンパイルを行います。

## 7. ライブラリ関数

本機で用意されているライブラリ関数を説明します。本機ではライブラリ関数のためのヘッダファイルは必要ありません。

本機では、標準入力装置・出力装置 (stream) は次のように、各機器に割り当てられています。

```
入力 stream : stdin (キーボード)
出力 stream : stdout (画面)、または stdprn (プリンタ)
S I O stream : stdaux (11ピン 半二重通信)
            stream : stdaux 1 (11ピン 全二重通信)
```

また、あらかじめ次のように # define されています。

```
# define N U L L      0
# define E O F        - 1
# define F I L E      int
```

本機では入力のためのライブラリ関数において、(SHIFT) (または (2nd F)) +  の入力は、それぞれ次の値を返します。

```
getch          : 0 x F F
その他の入力関数 : E O F
```

ラムデータファイル、S I O の入力関数では次のコードを区切り文字とします。

```
行の区切り : 0 x 0 d、0 x 0 a または 0 x 0 d + 0 x 0 a
ファイルの終了 : 0 x 1 a
(0 x 0 d + 0 x 0 a は入力時に 0 x 0 a に変換されます。)
```

(注) 区切りコードはできるだけ 0 x 0 a を使用してください。

ラムデータファイル、S I O の出力関数では次のコードを区切り文字とします。

```
行の区切り : ヌル
(0 x 0 a は出力時に 0 x 0 d + 0 x 0 a に変換されます。)
```

関数形式、説明文の記述において、[ ] で囲まれた記述はオプションで、記述の省略が可能です。

### 7. 1 標準入出力関数

#### ● getch, getchar, fgetc

```
形式:  int getch (F I L E * stream);
        int getchar (void);
        int fgetc (F I L E * stream);
```

機能: 1 文字を読み込みます。stream が stdin のときは  を押すと文字を読み込みます。

getc 入力 stream から読み込みます。

getchar stdin から読み込みます。



fgetc 入力 stream から読み込みます。



戻り値: 読み込んだ文字を返します。

## ● gets, fgets

形式: char\* gets (char\* s);  
char\* fgets (char\* s, int n, FILE\* stream);

機能: 文字列を読み込みます。文字列を s に読み込むとき、オーバーフローチェックは行いません。

gets stdin から  の入力で終わる文字列を s に読み込みます。文字列の終わりの  はヌル文字 '¥0' に置き換えられます。

fgets 入力 stream から文字列を s に読み込みます。読み込みは、stdin では (n - 1) 個の文字を読むか、 を読み込むことで終わります。それ以外の stream では、(n - 1) 個の文字を読むか、行の区切り文字を読み込むことで終わります。文字列の最後 ( を入力したときはその後) にヌル文字 '¥0' が付加されます。

戻り値: 読み込んだ文字列 s を返します。EOF のときはヌルを返します。

## ● scanf, fscanf, sscanf

形式: int scanf (const char\* format [, address, ...]);  
int fscanf (FILE\* stream, const char\* format [, address, ...]);  
int sscanf (char\* s, const char\* format [, address, ...]);

機能: 書式付き入力を行います。


scanf stdin から読み込みます。

fscanf 入力 stream から読み込みます。

sscanf 文字列 s から読み込みます。

読み込まれた文字の並びを、書式を与える文字列 format で指示されたデータ型に変換し、対応する引数 address, ... の示すアドレスに格納します。

書式で与えられた変換指定の数と、引数で示すアドレスの数は同じでなければなりません。多すぎた場合は無視されますが、不足した場合、結果は保証されません。

scanf と fscanf は  を押すと入力されます。sscanf はヌル文字 '¥0' が文字列の終了とみなされます。




scanf、fscanf は次の場合に処理を終了します。

- 1) フォーマット終了 (文字列終了マーク '¥0' の読み込み)
- 2) 読み込みデータとフォーマットの不一致
- 3) 読み込み終了

戻り値: 正しく変換、入力したデータの数を返します。入力データがない場合は 0 を返します。

EOF のとき、あるいはエラーのときは -1 を返します。

## ■ 入力の書式指定

入力データの書式を与える文字列は、書式指定、空白文字、非空白文字で構成されます。空白文字は空白 ()、タブ ()、 です。非空白文字は空白文字、% 文字以外の全ての文字です。

scanf () は次のように動作します。

- 書式文字列中に空白文字があると、入力における空白文字を、次に非空白文字が現われるまで読みとばします。

- 書式文字列中に非空白文字があると、それに一致する文字を読み込みます。不一致であれば、不一致文字は読み込まないで終了します。

- 書式指定は入力フィールドを読んで指定に従って変換し、引数で指定されるアドレスへ格納します。たとえばコンマ , を区切りとするには次のように書式指定します。

```
scanf ('%d, %d, %d', &a, &b, &c);
```

入力データの形式は次のようになります。

```
1 2, 3 4, 5 6 
```

## ◆ 変換指定

入力された文字の並びをどのようなデータ型として解釈するか、つまりどのように変換するかを指示するのが変換指定です。変換指定は % で始まり変換文字で終わる綴りです。

| 変換指定 | 期待される入力の形式           | 変換の形式            |
|------|----------------------|------------------|
| %d   | 10進整数形式の文字列          | int に変換          |
| %i   | 10 / 8 / 16進整数形式の文字列 | "                |
| %o   | 8進整数形式の文字列           | "                |
| %u   | 符号なし10進整数形式の文字列      | unsigned int に変換 |
| %x   | 16進整数形式の文字列          | int に変換          |
| %f   | 浮動小数点数形式の文字列         | float に変換        |
| %e   | "                    | "                |
| %g   | "                    | "                |
| %c   | 1文字がそのまま入力となる        | char に変換         |
| %s   | 文字列がそのまま入力となる        | 文字列の最後に '¥0' がつく |
| %p   | 16進4文字の文字列 (89ab など) | ポインタに変換          |

## ◆ 変換のオプション指示

%記号と変換文字の間にオプションの指示を与えることができます。

%[\*] [入力幅] [変換文字の修飾] 変換文字

[ ] で囲んだ各要素がオプション指示で、必要に応じて指定します。

## ① 代入抑制文字

- \* 変換指示に対応するフィールドは読みとばされ、代入されません。

## ② 入力幅

- n 変換する文字数を指定します。

指定したフィールド幅の中に空白文字があった場合、空白文字までが入力されます。

省略 空白文字、または変換できない文字までが入力幅となります。

## ③ 変換文字の修飾

- l (エル) 変換文字が整数への変換を指示している場合、long int に変換します。  
変換文字が実数への変換を指示している場合、double に変換します。
- L 変換文字が実数への変換を指示している場合、long double に変換します。

### ● putc, putchar, fputc

形式: `int putc (int c, FILE* stream);`  
`int putchar (int c);`  
`int fputc (int c, FILE* stream);`

機能: 1文字を出力します。  
       putc      出力 stream に1文字を出力します。  
       putchar   stdout に1文字を出力します。  
       fputc      出力 stream に1文字を出力します。

戻り値: 書き込まれた文字を返します。エラーが発生した場合は、EOFを返します。

### ● puts, fputs

形式: `int puts (const char* s);`  
`int fputs (const char* s, FILE* stream);`

機能: 文字列を出力します。  
       puts      ヌル文字 '¥0' で終わる文字列 s を、ヌル文字を改行文字に置き換えて stdout  
                   に出力します。  
       fputs      ヌル文字 '¥0' で終わる文字列 s を出力 stream に出力します。改行文字はつ  
                   けず、最後のヌル文字は出力されません。

戻り値: 負でない値を返します。エラーが発生した場合は、負の値を返します。

### ● printf, fprintf, sprintf

形式: `int printf (const char* format [, arg, ...]);`  
`int fprintf (FILE* stream, const char* format [, arg, ...]);`  
`int sprintf (char* s, const char* format [, arg, ...]);`

機能: 引数 arg, ... によって与えられる数値や文字、文字列を、書式文字列 format に従っ  
       て変換された文字の並びとして、出力します。  
       format 中の文字はそのまま出力されますが、%で始まる書式制御文字列は、それに対応  
       する引数の書式制御に使用されます。format で指定した書式制御文字列の型と引数の型が  
       対応していなかったり、不足している場合は、出力は保証されません。引数の数が書式文字  
       列より多い場合には、余った引数は無視されます。  
       printf     stdout に出力します。  
       fprintf    出力 stream に出力します。  
       sprintf    文字列 s に書き込みます。最後にヌル文字 '¥0' を付加します。

戻り値: 出力した文字数を返します。エラーが発生した場合は、負の数が返されます。

### ■ 出力の書式指定

printf は書式文字列に従って処理されます。書式文字列中に変換指示があると、対応する引数を指定  
       に従って変換して出力します。変換指定でない文字列はそのまま出力されます。

### ◆ 変換指定

| 変換指定 | 引数の種類  | 変換の形式                              |
|------|--------|------------------------------------|
| %d   | int    | 符号付き10進数で表示                        |
| %i   | int    | 符号付き10進数で表示                        |
| %o   | int    | 符号なし8進数で表示                         |
| %u   | int    | 符号なし10進数で表示                        |
| %x   | int    | 符号なし16進数で表示 (abcdef を使う)           |
| %X   | int    | ” (A B C D E F を使う)                |
| %f   | double | [−] ddd.ddd の形の10進数で表示 (d は10進数1桁) |
| %e   | double | [−] d. ddde ± dd の指数形式で表示          |
| %E   | double | [−] d. dddE ± dd の指数形式で表示          |
| %g   | double | 上記の f または e のうち、短い方で表示             |
| %G   | double | 上記の f または E のうち、短い方で表示             |
| %c   | int    | 1文字として表示 (int は unsigned char に変換) |
| %s   | 文字列    | 引数で指し示される文字列を表示                    |
| %p   | ポインタ   | 引数をポインタとして表示                       |

### ◆ 変換指定のオプション指示

変換指定は%で始まり、変換文字で終わる綴りですが、%と変換文字の間にはオプション指示を与える  
       ことができます。変換指示子の一般的な形式は次のようになります。

% [フラグ] [印字幅] [ , 精度] [サイズ] 変換文字

[ ] で囲んだ各要素がオプション指示で、必要に応じて使うことになります。また記述する順序も上  
       の形式に従わなければなりません。それぞれのオプションを説明しましょう。

#### ① フラグ

- − 変換の結果は左詰めで表示されます。
- + 符号付きは必ず+または−符号から始まります。
- # 変換文字が o の場合は先頭が 0 (ゼロ) で始まります。  
   x (または X) の場合は 0x (または 0X) で始まります。
- 0 数値型文字のフィールド内をスペースの代わりに 0 で埋めます。ただし、d、i、o、u、x、X  
   で精度指定がある場合は、0 フラグは無効です。0 とーの両方があれば、−の処理になります。
- 省略 右詰めで出力されます。

#### ② 印字幅

- n 変換された結果を表示する桁数 (フィールド幅) の指定です。  
   出力の幅がフィールド幅より少ない場合、残りは空白で埋められます。  
   たとえば %10d と指定すると10文字分の幅で10進数として出力されます。
- 0n n と同じくフィールド幅を指定します。  
   出力の値が n より小さい場合は先頭から 0 で埋められます。
- 省略 必要なフィールド幅がとられます。

#### ③ 精度指定

- n 浮動小数点数の出力で小数点以下の桁数を指定します。(最後の桁は丸められる)
- 省略 変換文字が e、E、f では小数点以下は6桁。  
   変換文字が g、G では有効数字がすべて出力されます。



## ④サイズ

l (エル) l (エル) 指示で d、i、o、u、x、X は long 型、n は long 型へのポインタになります。

## ● fflush

形式: int fflush (FILE\* stream);

機能: stream のバッファをフラッシュします。

入力 stream の場合はバッファの内容がクリアされます。その他は何もしません。stream がヌルのときは、全 stream に対してフラッシュします。

戻り値: 0 を返します。

## ● clearerr

形式: void clearerr (FILE\* stream);

機能: stream に関するエラーと EOF フラグをクリアします。

戻り値: ありません。

## 7. 2 文字処理関数

## ● isalnum, isalpha, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit

形式: int isalnum (int c);  
int isalpha (int c);  
int iscntrl (int c);  
int isdigit (int c);  
int isgraph (int c);  
int islower (int c);  
int isprint (int c);  
int ispunct (int c);  
int isspace (int c);  
int isupper (int c);  
int isxdigit (int c);

機能: 文字 c の分類をします。

isalnum 英字であるか、または数字であるかを判定します。

isalpha 英字であるかを判定します。

iscntrl 制御文字であるかを判定します。

(0x00~0x1f, 0x7f を真と判定)

isdigit 数字であるかを判定します。

isgraph 図形(グラフ)文字かを判定します。

(0x21~0x5f, 0x61~0x7e, 0x81~0x9f, 0xa1~0xf7 を真と判定)

islower 英小文字であるかを判定します。

isprint 印字可能な文字であるかを判定します。

(0x20~0x7e, 0x80~0xf7 を真と判定)

ispunct 区切り文字であるかを判定します。

(0x21~0x2f, 0x3a~0x40, 0x5b~0x5f, 0x7b~0x7e, 0x81~0x9f, 0xa1~0xf7 を真と判定)

isspace 空白文字(空白、タブ、復帰、改行など)であるかを判定します。

(0x09~0x0d, 0x20 を真と判定)

isupper 英大文字であるかを判定します。

isxdigit 16進法の表記文字であるかを判定します。

戻り値: 判定が真の場合は 0 以外の値を返します。偽の場合は 0 を返します。

## ● tolower, toupper

形式: int tolower (int c);

int toupper (int c);

機能: tolower 英文字を小文字に変換します。

toupper 英文字を大文字に変換します。

戻り値: 変換した文字を返します。

## 7. 3 文字列処理関数

## ● strcat

形式: char\* strcat (char\* s1, const char\* s2);

機能: 文字列 s2 を文字列 s1 の最後につけ加えます。付加したときオーバーフローチェックは行いません。

戻り値: 連結された文字列へのポインタを返します。

## ● strchr

形式: char\* strchr (const char\* s, int c);

機能: 文字列 s を先頭から調べて、文字 c を探します。

戻り値: 最初に見つかった文字 c へのポインタを返します。

見つからなかったときは NULL ポインタを返します。

## ● strcmp

形式: int strcmp (const char\* s1, const char\* s2);

機能: 文字列 s1 と s2 を比較します。各文字列の先頭から順に、異なる文字が現われるまで、または文字の終わりまで 1 文字ずつ比較していきます。

戻り値: 比較の結果を次の値で返します。

負数 s1 が s2 より小さい場合

0 s1 と s2 が同じ場合

正数 s1 が s2 より大きい場合

## ● strcpy

形式: char\* strcpy (char\* s1, const char\* s2);

機能: 文字列 s2 を、文字列 s1 にコピーします。コピーしたときオーバーフローチェックは行いません。

戻り値: 文字列 s1 へのポインタを返します。

## ● strlen

形式: int strlen (const char\* string);

機能: 文字列 string の長さを計算します。文字列の終わりの ' \0 ' は数えません。

戻り値: string 中の文字数を返します。

## 7. 4 メモリ割り当て関数

### • calloc

形式: void\* calloc (unsigned n, unsigned size);

機能: size バイトの大きさのメモリ領域をn個割り当てます。

戻り値: 割り当てられた領域のポインタを返します。割り当てに失敗した場合はNULLを返します。

(注) 確保サイズは、RAMサイズを超えないようにしてください。RAMサイズを超える値を指定した場合の戻り値は保証されません。

### • malloc

形式: void\* malloc (unsigned size);

機能: size バイトの大きさのメモリ領域を割り当てます。

戻り値: 割り当てられた領域のポインタを返します。割り当てに失敗した場合はNULLを返します。

### • free

形式: void free (void\* ptr);

機能: calloc, malloc で確保されたメモリ領域を解放します。

ptr は、calloc, malloc により割り当てられたメモリ領域のポインタでなければなりません。

戻り値: ありません。

## 7. 5 数学関数

### • abs

形式: int abs (int x);

機能: 整数の絶対値を求めます。

戻り値: 0 ~ 32767 の整数を返します。

### • asin, acos, atan

形式: double asin (double x);

double acos (double x);

double atan (double x);

機能: x に対する逆三角関数の値 (角度) を計算します。

扱う角度の単位は angle 関数で (DEG: 度)、(RAD)、(GRAD) を指定できます。

asin, acos では、引数 x の値は -1 ~ 1 の間でなければなりません。

戻り値: 次の表の範囲の値を返します。

| 逆三角関数 | 逆三角関数の値の範囲 |                     |              |
|-------|------------|---------------------|--------------|
|       | DEG        | RAD                 | GRAD         |
| asin  | -90° ~ 90° | $-\pi/2 \sim \pi/2$ | -100° ~ 100° |
| acos  | 0° ~ 180°  | $0 \sim \pi$        | 0° ~ 200°    |
| atan  | -90° ~ 90° | $-\pi/2 \sim \pi/2$ | -100° ~ 100° |

### • asinh, acosh, atanh

形式: double asinh (double x);

double acosh (double x);

double atanh (double x);

機能: 逆双曲線関数の値を計算します。

戻り値: 計算の結果を返します。

### • exp

形式: double exp (double x);

機能: 指数関数 ( $e^x$ ) を計算します。ただし e は自然対数の底です。

戻り値:  $e^x$  を返します。

### • log, log10

形式: double log (double x);

double log10 (double x);

機能: log x の自然対数を計算します。

log10 x の常用対数を計算します。

戻り値: 計算結果を返します。

### • pow

形式: double pow (double x, double y);

機能: x の y 乗を計算します。

戻り値: 計算結果を返します。

### • sin, cos, tan

形式: double sin (double x);

double cos (double x);

double tan (double x);

機能: x に対する三角関数の値を求めます。

扱う角度の単位は angle 関数で (DEG: 度)、(RAD)、(GRAD) を指定できます。

戻り値: 計算の結果を返します。

### • sinh, cosh, tanh

形式: double sinh (double x);

double cosh (double x);

double tanh (double x);

機能: 双曲線関数の値を計算します。

戻り値: 計算の結果を返します。

### • sqrt

形式: double sqrt (double x);

機能: x の平方根を計算します。

戻り値: 計算の結果を返します。

## 7. 6 ハードウェア・インタフェース関数

I/Oポート関数、機械語インタフェース関数は、誤って使うとプログラムやシステムエリアなどを破壊することがありますので、十分注意してください。

## ■ミニI/O関数

### ●miniget

形式: int miniget (void);

機能: ミニI/Oから1バイトのデータを読み込みます。

ビット2: Xin    ビット1: Din    ビット0: Ack

戻り値: 入力データを返します。

### ●miniput

形式: void miniput (char byte);

機能: ミニI/Oに1バイトのデータを出力します。

ビット2: Busy    ビット1: Dout    ビット0: Xout

戻り値: ありません。

## ■11ピンの8ビット制御関数

11ピンで8ビット制御を可能にします。

### ●fclose

形式: int fclose (FILE \* stream);

機能: streamで指定されたファイルを閉じます。

戻り値: 正常に終了したときは0を返し、エラーになったときはEOFを返します。

### ●fopen

形式: FILE \*fopen (char \*path, char \*type);

機能: 11ピンを8ビット制御モードに設定します。

pathでデバイス名を、typeでモードを指定します。

'pio': 8ビット制御    'r+', 'w+'または'a+': データの入出力

戻り値: 正常に終了したときはオープンしたストリームへのポインタを返します。

エラーになったときはヌルを返します。

### ●pioget

形式: int pioget (void);

機能: 11ピンから8ビットデータを入力します。

戻り値: 入力データを返します。

### ●pioput

形式: void pioput (char byte);

機能: 11ピンへ8ビットデータを出力します。

戻り値: ありません。

### ●pioset

形式: void pioset (char byte);

機能: 8ビット制御の入出力モードを設定します。

1のときは入力モードの指定になり、0のときは出力モードの指定になります。

戻り値: ありません。

## ■SIO (RS-232C) 制御関数

### ●fclose

形式: int fclose (FILE \* stream);

機能: streamで指定されたファイルを閉じます。

全二重モードで使用しているときは、テキスト終了コード (TEXTモードのFORMATで指定している終了コード) を書き込みます。

戻り値: 正常に終了したときは0を返し、エラーになったときはEOFを返します。

### ●fopen

形式: FILE \*fopen (char \*path, char \*type);

機能: 11ピンをRS-232C準拠で通信を行うモードに設定します。

(通信条件の設定は、TEXTモードのSIOで行ってください。)

デバイス名 (pathで指定)

'stdaux': 半二重通信の指定    'stdaux1': 全二重通信の指定

アクセスモード (typeで指定)

'r+', 'w+'または'a+': データの入出力の指定

戻り値: 正常に終了したときはオープンしたストリームへのポインタを返します。

エラーになったときはヌルを返します。

(注) fopen()を使用する場合は、必ずfclose()で終了してください。fclose()を使用しない場合、送信データおよび終了コードが最後まで正しく送信されません。この場合、電源をオフにしたり、モードを切り替えても送信されません。

## ■通信バッファ制御

### ●feof

形式: int feof (FILE \* stream);

機能: 通信バッファのデータの有無を調べます。

戻り値: データが最後に達していれば-1を返し、そうでなければ0を返します。

## ■I/Oポート関数

### ●inport

形式: unsigned char inport (unsigned char port);

機能: アドレスportのI/Oポートから1バイトのデータを読み込みます。

本機では0x20から0x3FのI/Oポートが使用できます。

戻り値: 入力データを返します。

### ●outport

形式: void outport (unsigned char port, unsigned char byte);

機能: アドレスportのI/Oポートに1バイトのデータを出力します。

本機では0x20から0x3FのI/Oポートが使用できます。

戻り値: ありません。

## ■機械語インタフェース関数

### ●call

形式: unsigned call (unsigned adr, void \* arg\_HL);

機能： アドレス `adr` から始まる機械語を呼び出します。引数 `arg__HL` の値が `HL` レジスタに渡されます。

戻り値： `HL` レジスタの値が返されます。

#### ● peek

形式： `unsigned char peek (unsigned adr);`

機能： アドレス `adr` から1バイトを読み込みます。

戻り値： 読み込んだデータを返します。

#### ● poke

形式： `void poke (unsigned adr, unsigned char byte);`

機能： アドレス `adr` に1バイトデータ `byte` を書き込みます。

戻り値： ありません。

## 7. 7 データファイル関数

#### ● fclose

形式： `int fclose (FILE * stream);`

機能： `stream` で指定されたファイルを閉じます。

'w' または 'a' モードでオープンしているときは、終了コード (1A) を書き込みます。

戻り値： 正常に終了したときは0を返し、エラーになったときはEOFを返します。

#### ● feof

形式： `int feof (FILE * stream);`

機能： ファイルの終わりを調べます。

戻り値： ファイルの終わりを検出したときは-1を返し、検出できなかったときは0を返します。

#### ● flob

形式： `unsigned long flob (FILE * stream);`

機能： ファイルの未使用バイト数を求めます。

戻り値： 残りバイト数を返します。

#### ● fopen

形式： `FILE *fopen (char *path, char *type);`

機能： データファイルにファイル番号を割り当て、オープンモードを指定し入出力を可能にします。  
アクセスモード (type で指定)

'r' : データの読み出しの指定

'w' : データの書き込みの指定

'a' : データの追加書き込みの指定

戻り値： 正常に終了したときはオープンしたストリームへのポインタを返します。  
エラーになったときはヌルを返します。

## 7. 8 グラフィック関数

表示用のグラフィック関数です。機能はBASICでの命令と同じです。引数などの詳細についてはBASICの各命令の説明を参照してください。

#### ● circle

形式： `int circle (int x, int y, int r, double s-angle, double e-angle, double ratio, int reverse, unsigned short kind);`

機能： 円を描き、内部を塗りつぶします。開始角度から終了角度までを反時計方向に円を描きます。開始角度、終了角度は10進数で指定します。

x, y : 円の中心点

r : 半径

s-angle : 開始角度

e-angle : 終了角度

ratio : 比率 (Y軸方向の半径/X軸方向の半径)

reverse : ドットの種類

0 : ドットをセット

1 : ドットをリセット

2 : ドットの反転

kind : 塗りつぶす模様の種類 BASIC命令と同じ

(第10章 BASICの各命令の説明を参照)

戻り値： 正常に終了したときは0を返し、エラーになったときは-1を返します。

(注) 半径線を描くときは、開始角度や終了角度をマイナスで指定します。

0°の位置に半径線を描くときは、-360°と指定します。

#### ● gcursor

形式： `int gcursor (int x, int y);`

機能： グラフィックの表示開始位置をドット (点) 単位で指定します。

戻り値： 正常に終了したときは0を返し、エラーになったときは-1を返します。

#### ● gprint

形式： `void gprint (char * image);`

機能： 指定されたドットパターンを表示します。

戻り値： ありません。

#### ● line

形式： `int line (int x1, int y1, int x2, int y2, int reverse, unsigned short mask, int rectangle);`

機能： 指定された2点間を線で結びます。



reverse :

- 0 : ドットをセット
- 1 : ドットをリセット
- 2 : ドットの反転

mask : BASIC命令と同じ

(第10章 BASICの各命令の説明を参照)

rectangle :

- 0 : 指定した mask で直線を描きます。
- 1 : 指定した mask で長方形を描きます。
- 2 : 指定した mask で長方形の内部を塗りつぶします。

戻り値 : 正常に終了したときは0を返し、エラーになったときは-1を返します。

#### ● paint

形式 : int paint (int x, int y, unsigned short kind);

機能 : 囲まれた領域を塗りつぶします。

x, y : 指定した点の座標

kind : 塗りつぶす模様の種類 BASIC命令と同じ

(第10章 BASICの各命令の説明を参照)

戻り値 : 正常に終了したときは0を返し、エラーになったときは-1を返します。

#### ● point

形式 : int point (int x, int y);

機能 : 指定したドットの状態を読み取ります。

戻り値 : 指定したドットが点灯しているときは1を返し、消灯しているときは0を返します。指定したドットが画面外になるときは-1を返します。

#### ● preset

形式 : int preset (int x, int y);

機能 : 指定したドットを消します。

戻り値 : 正常に終了したときは0を返し、エラーになったときは-1を返します。

#### ● pset

形式 : int pset (int x, int y, int reverse);

機能 : 指定したドットの点灯または反転を行います。

reverse :

- 0 : ドットを点灯
- 1 : ドットを反転

戻り値 : 正常に終了したときは0を返し、エラーになったときは-1を返します。

## 7. 9 その他の関数

#### ● abort, exit

形式 : void abort (void);

void exit (int status);

機能 : 現在実行しているプログラムを終了します。

abort プログラムを異常終了させます。画面に **ABORT** が表示されます。

exit プログラムを正常終了させます。画面に **EXIT** に続いて status の値が表示されます。

戻り値 : ありません。

#### ● angle

形式 : void angle (unsigned n);

機能 : n の値によって、三角関数の単位が設定されます。

n = 0 60分法 : 単位 DEG (度)

n = 1 弧度法 : 単位 RAD (ラディアン)

n = 2 グラード法 : 単位 GRAD (グラード)

戻り値 : ありません。

#### ● breakpt

形式 : void breakpt (void);

機能 : プログラムの実行を中断して、ブレイクモードに入ります。

戻り値 : ありません。

#### ● clrscr


形式 : void clrscr (void);

機能 : 画面をクリアし、カーソル (表示位置) を (0, 0) にセットします。

戻り値 : ありません。

#### ● getch

形式 : int getch (void);

機能 : 1文字を stdin から直接読み込みます。読み込みに  の入力はありません。入力バッファが空のときは、次の入力があるまで待機します。読み込んだ文字は表示されません。

戻り値 : 読み込んだ文字を返します。

#### ● gotoxy

形式 : void gotoxy (unsigned x, unsigned y);

機能 : 画面のカーソルを、座標 (x, y) に移動します。

ただし、左上隅の座標を (0, 0) とします。

戻り値 : ありません。

#### ● kbhit

形式 : int kbhit (void);

機能 : キーボードからのキー入力の有無を調べます。

戻り値 : キー入力が行われていれば0以外の値を返し、行われていなければ0を返します。

## 8. エラーメッセージ

### 8. 1 コンパイルエラーメッセージ

| エラーメッセージ                      | エラーの内容                                                                                              |
|-------------------------------|-----------------------------------------------------------------------------------------------------|
| Null dimension                | 配列宣言で要素数が書かれていないものがある                                                                               |
| array of function is illegal  | 要素が関数の配列を宣言している                                                                                     |
| can't find include file       | インクルードファイルが見つからない                                                                                   |
| case not in switch            | caseラベルがswitch文の中でない                                                                                |
| constant expected             | <ul style="list-style-type: none"> <li>● 配列宣言で要素数が整数定数式でない</li> <li>● caseラベルの式が整数定数式でない</li> </ul> |
| default not in switch         | defaultラベルがswitch文の中でない                                                                             |
| define buffer full            | #defineが多すぎる                                                                                        |
| different s/u                 | 構造体／共用体の代入で両辺の型が違う                                                                                  |
| division by 0                 | ／または%演算子の右辺が0である                                                                                    |
| duplicate # define : 名前       | 定義済みのマクロ名を定義している                                                                                    |
| duplicate case                | switch文中に同じ値のcaseラベルがある                                                                             |
| duplicate default             | switch文中にdefaultラベルが複数ある                                                                            |
| duplicate label : 名前          | 定義済みのgotoラベルを定義している                                                                                 |
| empty character constant      | 文字定数構文中に文字が1つもない                                                                                    |
| float overflow                | 実数の定数値が大きすぎる                                                                                        |
| float underflow               | 実数の定数値が小さすぎる                                                                                        |
| function illegal in s/u       | 構造体／共用体の中で関数を宣言している                                                                                 |
| function returns illegal type | 関数値として返せない型を定義している                                                                                  |
| if nest too deep              | # if / # ifdef のネスティングが深すぎる                                                                         |
| if nesting error              | # if / # ifdef と # endif の対応が取れていない                                                                 |
| if -less elif                 | # elif に対応する # if / # ifdef がない                                                                     |
| if -less else                 | # else に対応する # if / # ifdef がない                                                                     |
| if -less endif                | # endif に対応する # if / # ifdef がない                                                                    |
| illegal # line                | # define 行の構文が不正である                                                                                 |
| illegal break                 | break文がdo、for、while、switch文の中でない                                                                    |
| illegal character             | ソースプログラムに不正文字がある                                                                                    |
| illegal class                 | 記憶クラスが不正である                                                                                         |
| illegal continue              | continue文がdo、for、while文の中でない                                                                        |
| illegal digit in octal        | 8進数内に不正数字(8と9)がある                                                                                   |
| illegal function              | 関数型でないもので関数呼び出しを行っている                                                                               |
| illegal if                    | # if / # ifdef 行の構文が不正である                                                                           |
| illegal include               | # include 行の構文が不正である                                                                                |
| illegal indirection           | 単項*演算子のオペランドが不正である                                                                                  |

| エラーメッセージ                      | エラーの内容                                                                                                       |
|-------------------------------|--------------------------------------------------------------------------------------------------------------|
| illegal initialization        | 初期化の右辺が定数式でない                                                                                                |
| illegal main                  | main関数に引数を宣言している                                                                                             |
| illegal operand of 演算子        | 演算子のオペランドの型が不正である                                                                                            |
| illegal operand of U+         | 単項+演算子のオペランドの型が不正                                                                                            |
| illegal operand of U-         | 単項-演算子のオペランドの型が不正                                                                                            |
| illegal operand of ARG        | 関数呼び出しの引数の型が不正                                                                                               |
| illegal operand of RET        | リターン文の式の型が不正                                                                                                 |
| illegal s/u                   | 構造体／共用体が不正使用されている                                                                                            |
| illegal size                  | 構造体／共用体のサイズが大きすぎる                                                                                            |
| illegal switch expression     | switch(e)の式のeの型が不正である                                                                                        |
| illegal type                  | 不正な型変換が発生している                                                                                                |
| illegal void                  | void型の使用が正しくない                                                                                               |
| include nest too deep         | # include のネスティングが深すぎる                                                                                       |
| macro recursion               | マクロが再帰している                                                                                                   |
| memory full                   | メモリが不足している                                                                                                   |
| missing argument : 名前         | 関数定義の( )内には引数を宣言している                                                                                         |
| missing declarator            | 宣言子がない                                                                                                       |
| missing function : 名前         | 使用された関数が定義されていない                                                                                             |
| missing label                 | <ul style="list-style-type: none"> <li>● goto文にラベルがない</li> <li>● 使用されたgotoラベルが定義されていない</li> </ul>            |
| missing main                  | main( )が定義されていない                                                                                             |
| missing member                | <ul style="list-style-type: none"> <li>● メンバー未定義の構造体／共用体を初期化している</li> <li>● 式中で定義していないメンバーを使用している</li> </ul> |
| missing member in s/u         | 構造体／共用体の定義でメンバーが1つもない                                                                                        |
| missing name in prototype     | 関数定義のプロトタイプに引数名がない                                                                                           |
| missing type                  | 型を宣言していない                                                                                                    |
| missing type in prototype     | プロトタイプで構文規則違反がある                                                                                             |
| newline in character constant | 文字定数の構文内に改行文字がある                                                                                             |
| newline in string constant    | 文字列定数の構文内に改行文字がある                                                                                            |
| prototype unmatched           | 関数呼び出しの式がプロトタイプに適合しない                                                                                        |
| redeclaration : 名前            | 定義済みの名前を定義している                                                                                               |
| reserved : 名前                 | この名前は予約されているので使用禁止                                                                                           |
| syntax error                  | 構文規則に適合していない                                                                                                 |
| token buffer full             | マクロ展開が複雑すぎる                                                                                                  |
| too complicated declarator    | 宣言が複雑すぎる                                                                                                     |
| too complicated declaration   | 宣言が複雑すぎる                                                                                                     |
| too complicated expression    | 式が複雑すぎる                                                                                                      |
| too complicated initialize    | 初期化が複雑すぎる                                                                                                    |
| too deep statement            | 文のネスティングが深すぎる                                                                                                |

| エラーメッセージ                                     | エ ラ ー の 内 容              |
|----------------------------------------------|--------------------------|
| too long initializer                         | 初期化で文字列定数が長すぎる           |
| too long macro                               | マクロ本文が長すぎる               |
| too many #define                             | #defineの個数が制限を超えている      |
| too many case                                | caseラベルの個数が制限を超えている      |
| too many characters<br>in character constant | 文字定数の文字数が制限を超えている        |
| too many characters<br>in string constant    | 文字列定数の文字数が制限を超えている       |
| too many initializers                        | 初期化式の個数が宣言より多い           |
| too many label                               | gotoラベルの個数が制限を超えている      |
| too many prototype                           | プロトタイプ個数が制限を超えている        |
| unacceptable operand of &                    | &演算子のオペランドが不正である         |
| undefine: 名前                                 | 未定義の名前を使用している            |
| unexpected EOF                               | 構文規則の途中でソースプログラムが終了している  |
| unknown size                                 | サイズが確定できない               |
| void function                                | void関数なのにreturn文で値を返している |
| zero or negative subscript                   | 配列の要素数が0または負である          |

## 8. 2 実行時エラーメッセージ

| エラーメッセージ         | エ ラ ー の 内 容                                                                                                          |
|------------------|----------------------------------------------------------------------------------------------------------------------|
| NO MEMORY        | 関数の再帰などでメモリを使い果たした                                                                                                   |
| BAD POINTER      | ポインタでの代入、C領域以外のメモリへ書き込もうとした                                                                                          |
| DIVISION BY 0    | 0での割り算                                                                                                               |
| UNKNOWN ERROR    | ポインタでの代入で、C領域が破壊されている                                                                                                |
| BAD FUNCTION     | ポインタでの関数呼び出しで、ポインタ値がおかしい                                                                                             |
| BAD STREAM       | 入出力ライブラリのストリームが適正でない                                                                                                 |
| ARITHMETIC ERROR | <ul style="list-style-type: none"> <li>● 数学ライブラリのシステムコールでエラーリターンした</li> <li>● 浮動小数点数に関するシステムコールがエラーリターンした</li> </ul> |
| FRAME ERROR      | 関数フレームが破壊された                                                                                                         |
| I/O OPEN ERROR   | SIOデバイスをオープンしているのに再度オープンした                                                                                           |
| I/O ERROR        | ミニI/Oをオープンしていない                                                                                                      |

# 第7章 CASL

## 1. CASL (アセンブラ言語)

コンピュータ言語においてBASIC、FORTRAN、COBOLといった高水準な言語は、その記述がわかりやすいためプログラミングが容易であり、開発期間も短くて済みます。しかしこれらは、メモリの作業領域を多く必要としたり処理速度が遅いといった面があります。コンピュータの性能を最も効率よく活用できるのは機械語ですが、機械語コードで入力する必要がある、プログラミングは非常に手間がかかります。

アセンブラはこの機械語に最も近い言語です。BASICほど手軽ではないにしても、機械語に比べて使いやすく、コンピュータの性能を限界近くまで活用できます。ただ、アセンブラ言語で効率の良いプログラムを作るには、各命令を実行するときに、レジスタやフラグなどがどのような動きや働きをするのかを知る必要があります。これらの動き、働きなどはコンピュータのハードウェアに大きく左右され、メーカーや機種により異なるのが普通です。

このため経済産業省の情報処理技術者試験では、COMETという仮想の計算機を設定し、これのアセンブラ言語としてCASLの仕様を取り決めたうえで試験問題を出しています。(平成13年度より、計算機はCOMET IIに、アセンブラ言語はCASL IIに改定されていますが、本機はこれらに対応していません。)

本機のCASLモードは、このCASLの仕様に準拠したアセンブラ言語が扱えるため、情報処理技術者試験(基本情報技術者試験(旧第2種)では選択)のトレーニングとしてプログラミングからデバッグそしてシミュレーションまでの学習にお使いいただけます。

この取扱説明書は、本計算機の操作方法に重点をおいて説明しています。COMET IIやCASL IIの仕様、CASL IIの文法、プログラミングについての詳しい説明は、有名書店で販売されている情報処理技術者試験のCASL IIに関する書籍を参照してください。

COMETの仕様については、265ページを参照してください。

## 2. CASLモードの構成

CASLモードは次の3つの機能から構成されています。(246ページの機能一覧表を参照ください。)

**アセンブラ** : TEXTエディタで作成したソースプログラムをオブジェクトプログラムに変換してメモリに書き込みます。

CE-126P(プリンタ)をお持ちの場合は、CE-126Pが接続され、“PRINT”が点灯しているとき、アセンブルした結果を印字できます。





## 3. ソースプログラムの作成、編集（エディット）

CASLのソースプログラムの作成や変更などはTEXT（テキスト）モードのエディタ機能を用いて行います。

**TEXT** **E** と押して、テキストエディタを選んでください。

```
TEXT EDITOR
<
```

● TEXTモードのくわしい使いかたは、「第5章 TEXTモード」を参照してください。

### 3. 1 ソースプログラムの入力形式

ソースプログラムの構成を以下に示します。

```
3 2 7 6 7 BGN ADD 0, DAT, 1; SAMPLE
|-----|-----|-----|-----|
行番号欄 ラベル欄 命令コード欄 オペランド欄 注釈欄
(省略可) (省略可) (省略可) (省略可)
```

#### (1) 行番号（ラインナンバー）欄

- ① 行番号は1～65279までの数値を使用します。
- ② 1～65279の範囲外の値を指定すると“LINE NO. ERROR”が表示されます。

#### (2) ラベル欄

- ① ラベルは1～6文字まで使用できます。7文字以上を入力した場合は、先頭から6文字がラベルとして有効になります。また、79文字以上入力するとエラーになります。
  - ② CASLでは、ラベルは6文字以内で先頭の文字はアルファベット大文字、2文字目以降はアルファベット大文字または数字と定義されています。
- なお、ラベル欄にセミコロン（;）を書いて注釈行とすることもできます。
- ③ ラベルは行番号に続けて入力する必要があります。
  - ④ ラベル入力後、**TAB** で次の入力位置へカーソルを移し、命令コードを入力します。

\*\*\*\*\* **TAB** について \*\*\*\*\*

ラベル、命令コード、オペランドの間には1文字以上のスペースが必要です。なお、ラベルを省略する場合は、行番号の後ろに1文字以上のスペースを入れる必要があります。このとき、**SPACE** の代わりに **TAB** を使うと、命令コードやオペランドの表示位置がそろって見やすくなります。

#### (3) 命令コード欄

- ① アルファベットキーで命令コードを入力します。入力後、**TAB** で次の入力位置へカーソルを移します。

#### (4) オペランド欄

- ① オペランド欄はGRフィールド、adrフィールド、XRフィールドで構成されます。
- ② 各ブロック（フィールド）の区切りには、必ずコンマ（,）が必要です。
- ③ XRフィールドは省略することができます。

#### (5) 注釈欄

- ① プログラムに注釈を付加する場合は、オペランドの直後にセミコロン（;）を入力し、その後に注釈を入力します。

● 一行の長さは、注釈欄を含めて最大254文字までです。

### 3. 2 ソースプログラムの消去

TEXTモードの機能選択画面で、**D** を押せばデリート（Del）機能が選ばれ、次のようにテキスト内容を消去（削除）してよいかを聞いてきます。（テキスト内容がないときは、画面は変わりません。）

```
TEXT DELETE OK? (Y)
```

**Y** を押せば、テキスト内容がすべて消去され、TEXTモードの機能選択画面に戻ります。

**Y** 以外のキー（そのとき有効に働くキー）を押せば、消去されずに機能選択画面に戻ります。

### 3. 3 ソースプログラムの入力

TEXTモードの機能選択画面で **E** を押せばテキストエディタが選ばれます。

```
TEXT EDITOR
<
```

このとき、**▼** または **▲** を押すと、ソースプログラムなどのテキスト内容がある場合は、画面に表示されます。何もなしときは、画面は変わりません。

新たにソースプログラムを入力するときなどは、**BREAK ON** を押して機能選択画面に戻し、「3.2 ソースプログラムの消去」の方法でテキスト内容を消去してください。

次にソースプログラムの入力手順を説明します。

- ① 行番号を入力します。
- ② ラベルを入力するときは、行番号に続けて入力し **TAB** を押します。  
ラベルがないときはそのまま **TAB** を押します。カーソルが命令コード欄に移ります。
- ③ 命令コードを入力します。その後 **TAB** を押せばカーソルがオペランド欄に移ります。
- ④ オペランドは、コンマ（,）で区切って入力します。



## 4. 2 エラーメッセージ

アセンブルの実行で、ソースプログラムにエラーが検出されると、下表のようなエラーメッセージが表示されます。

エラーは **[CLS]** で解除して、TEXTエディタでソースプログラムを修正してください。

アセンブル時に使用するメモリは、オブジェクトを記憶するメモリを除いて、ラベル1つで8バイト、ワーク用として4バイト、スタック用として64バイトです。メモリが足りないとメモリエラーになります。

| エラーの種類   | エラーメッセージ            | 原因                                                                                                   |
|----------|---------------------|------------------------------------------------------------------------------------------------------|
| オペコードエラー | OP-CODE ERROR (行番号) | 行番号で示すプログラムの命令コードに誤りがある                                                                              |
|          | OP-CODE ERROR (0)   | ソースプログラムがない                                                                                          |
| オペランドエラー | OPERAND ERROR (行番号) | 行番号で示すプログラムのオペランドに誤りがある                                                                              |
| ラベルエラー   | LABEL ERROR (行番号)   | 行番号で示すプログラムのラベルに誤りがある                                                                                |
| メモリエラー   | MEMORY ERROR (0)    | <ul style="list-style-type: none"> <li>● オブジェクトを記憶するメモリが不足している</li> <li>● スタック用メモリが不足している</li> </ul> |
| その他のエラー  | OTHER ERROR         | プログラムの先頭にSTART命令がない、終わりにEND命令がないなど、入力形式が正しくない                                                        |

## 5. シミュレーション

メニュー画面で **[G]** を押すと、オブジェクトプログラムの実行処理に移ります。

**[G]**

```
<< SIMULATION >>
START ADDRESS=#1000
実行開始アドレス
```

実行開始アドレスは、START命令で指定されたアドレスが表示されます。START命令で実行開始アドレスが指定されていないときは、自動的に#1000が実行開始アドレスになります。

表示されている実行開始アドレスを変更する場合は、上記のアドレス表示状態で、変更したいアドレスを10進数、16進数、またはラベルで入力します。

次に **[F]** を押せば、実行方法選択画面になります。



```
<< SIMULATION >>
START ADDRESS=#1000

Normal      Trace
  ↑          ↑
ノーマル実行 トレース実行
```

**[N]** または **[T]** で選択します。

## 5. 1 ノーマル実行

実行方法選択画面で **[N]** を押せば、設定された実行開始アドレスからプログラムの実行を開始します。

**[N]**

```
♥♥CARDS♥♥

*** CASL ***
Assemble Monitor Go
```




EXIT命令が実行されると、メニュー画面に戻ります。

ノーマル実行中、次の場合はプログラムの実行を停止して、実行内容の表示や停止したアドレスを表示します。

- OUT命令を実行した場合。 **[F]** を押せばプログラムの継続実行ができます。
- ブレークポイント (BP) が設定されていて、プログラムカウンタ (PC) の値がブレークポイントの値と一致した場合。この場合、ブレークポイントが設定されているアドレスの命令は実行されています。
- **[BREAK]** が押された場合。この場合、プログラムカウンタに停止 (ブレーク) したときのアドレスが入っているので、 **[F]** を押せばプログラムの継続実行ができます。また、 **[CLS]** や **[BREAK]** でメニュー画面に戻ってから、再び実行操作をしても継続実行できます。
- \*命令を実行した場合 (263ページ参照)
  - \*命令により停止し、レジスタ内容などが表示されているときに **[F]** を押すと、\*命令をスキップして次の命令を実行します。
  - \*命令 (第1語目が#00XX、第2語目が#XXXXの命令語) を実行した場合、プログラムカウンタには第1語目のアドレスが入っています。したがって **[BREAK]** でプログラムを中断した後、オブジェクトの修正、または実行アドレスの修正を行わずにプログラムを再実行させると、再度同じアドレスでプログラムが停止します。このような場合は、命令語または実行アドレスを修正した後、プログラムを実行してください。
- 画面右下に“PRINT”が点灯しているときは、OUT命令による結果が印字されます。

## 5. 2 トレース実行

実行方法選択画面で **[T]** を押せば、表示されている実行開始アドレスからプログラムのトレース実行が開始されます。トレース実行中は1命令実行ごとに、実行結果 (GR0~GR4、PC、FRの内容) を表

示して停止します。ここで表示されるPCの値は命令実行後のPCの値、つまり次に実行する番地です。次に  を押すと、次の命令を実行して停止します。

```
1000: GR0:0000 GR4:1B0B
      GR1:0000 PC :1002
      GR2:0000 FR :0000
      GR3:0000 <PUSH>
```

(注) GR 4の値はメモリの使用状態により変わります。



```
1002: GR0:0000 GR4:1B0A
      GR1:0000 PC :1004
      GR2:0000 FR :0000
      GR3:0000 <PUSH>
```

実行した命令

メニュー画面で **[SHIFT]** + **[P<=>NP]** を押し、“PRINT” が点灯しているときは、以下のフォーマットで実行結果が（16進数で）印字されます。

〈印字例〉

```
ADD:GR0 GR1 GR2 GR3      ①アドレス
1000:0000 0000 0000 0000      ②GR 0
1002:0000 0000 0000 0000      ③GR 1
1004:0000 0000 0000 0000      ④GR 2
♥♥CARDS♥♥                  ⑤GR 3
0002:0000 0000 0000 0000
1006:0000 0000 0000 0000
1008:0000 0000 0000 0000

←①→←②→←③→←④→←⑤→
```

### 5. 3 シミュレーションでのエラー

シミュレーションでは、次のエラーが発生することがあります。

| エラーメッセージ     | 原因                                       |
|--------------|------------------------------------------|
| OBJECT ERROR | オブジェクトプログラムがない。                          |
| *MEM*        | ● J M P 命令などで使用可能な領域を超えたアドレスへのジャンプが行われた。 |
| *ERR*        | ● 使用可能な領域を超えて、データのロードやセーブをしようとした。        |
| *OPR*        | O U T 命令で出力文字を97文字以上にした。                 |
| *ERR*        |                                          |

## 6. モニタ

モニタ機能では、仮想計算機COMETの各レジスタの内容確認や、オブジェクトプログラムを表示できます。

また、レジスタの値やオブジェクトの設定・変更、およびシミュレーションのためのブレークポイントの設定などができます。

モニタ機能は、メニュー画面で **[M]** を押すことによって選べます。( **[SHIFT]** + **[ASMBL]** を押してから **[C]** を押すとメニュー画面になります。)

**[M]**

```
<< MONITOR >>
Register      Object
```

モニタ画面では、次の操作が可能です。

| キー操作       | 機能                                                      |
|------------|---------------------------------------------------------|
| <b>[R]</b> | レジスタの内容の表示と、レジスタへの値の設定ができます。                            |
| <b>[O]</b> | オブジェクトプログラムと逆アセンブルした命令の表示ができます。また、この内容を設定したり、変更したりできます。 |

### 6. 1 レジスタの内容の表示

モニタ画面で **[R]** を押すとレジスタの内容やブレークポイントの内容が表示されます。

**[R]**

```
GR0: #0000 0
GR1: #0000 0
GR2: #0000 0
GR3: #0000 0
GR4: #1B0B 6923
PC : #1000 4096
```

(注) GR 4の値はメモリの使用状態により変わります。

レジスタ 16進数で 10進数で  
内容を表示 内容を表示

以降、**[▼]** でカーソル（このときのカーソルはコロン（:）の消灯）を下に移動していけば、FR以降を画面に呼び出すことができます。

:



```
GR3: #0000 0
GR4: #1B0B 6923
PC : #1000 4096
FR : #0000 0
BP : #FFFF 65535
BC : #0000 0
```

戻すときは **[▲]** を押します。**[▼]** の代わりに **[↩]** を押しても同様に働きます。




各レジスタ名は以下のとおりです。

| 表 示       | レジスタ名     | 機 能                                                               |
|-----------|-----------|-------------------------------------------------------------------|
| GR 0～GR 4 | 汎用レジスタ    | 算術演算と論理演算に用います。GR 1～GR 4は指標レジスタとしても用います。また、GR 4はスタックポインタとしても用います。 |
| PC        | プログラムカウンタ | 次に実行すべき命令が格納されているアドレスを記憶します。                                      |
| FR        | フラグレジスタ   | 特定の命令を実行した結果が、正、ゼロ、負になると、それぞれ0、1、2にセットされます。                       |
| BP        | ブレークポイント  | シミュレーションでの実行を制御するために使用します。                                        |
| BC        | ブレークカウンタ  |                                                                   |


## 6. 2 レジスタに値を設定する方法


GR 0～GR 4、PC、FR、BP、BCの各レジスタに値を設定できます。

値を設定するときは、レジスタにカーソル（コロン（:）の消灯）を移し、そのまま、10進数、16進数、ラベル、文字などで入力して  を押します。

〈例〉10進数の入力： 1 2 3  # 0 0 7 B 1 2 3 が入力されます。



16進数の入力：# 0 0 7 B  # 0 0 7 B 1 2 3 が入力されます。

ラベルの入力： 'L 1'  ラベル 'L 1' が定義されているアドレスが入力されます。

文字の入力： 'A'  Aの文字コード # 0 0 4 1 6 5 が入力されます。

設定終了後、**(BREAK)** でモニタ画面に戻ります。

レジスタに値を設定するときは、次の点に注意してください。

- ①設定する値は10進数、16進数、ラベル、文字で入力します。
- ②負数は先頭に  でマイナスを入力します。
- ③設定できる値の範囲は-32768～65535までの整数です。この範囲を超えた値を入力して  を押すと、入力前の値に戻ります。
- ④FRの値は0、1、2だけです。0、1、2以外の値が入力されると、他のビットは無視されます。ただし、画面には入力した値が表示されます。
- ⑤アセンブル実行後の各レジスタには、以下の値が設定されています。

GR 0～GR 3： 0

GR 4： オブジェクトエリアの最上位番地+1

PC： メインプログラムの実行開始アドレス  
(STARTで指定されたラベルのアドレス)



FR： 0

BP： FFFF (16進) 6 5 5 3 5 (10進)  
(ブレークポイントが設定されていない状態)

BC： 0

(注) GR 4はスタックポインタとして最大限に使えるように、オブジェクトエリアの最上位番地+1に指定されています。この値はモニタ機能で呼び出してマニュアル操作で変更、またはプログラム上で変更できます。


## 6. 3 オブジェクトコードの表示

アセンブルの実行によって作成されたオブジェクトプログラムを表示させることができます。メニュー画面で  を押すとモニタ画面になり、モニタ画面で  を押すとオブジェクトプログラムが記憶されているときは、アドレス#1000が表示されます。



```
<< OBJECT >>

ADDRESS=#1000
メインプログラムの開始アドレス
```

次に  を押すと、オブジェクトエリアの#1000番地に記憶された内容から表示されます。表示内容は、オブジェクトコードと逆アセンブルしたニモニックです。










⋮




```
第1語目→ 1000 7000 PUSH #100B
第2語目→ 1001:100B
第1語目→ 1002:7000 PUSH #100A
第2語目→ 1003:100A
          1004:8000 CALL #0002
          1005:0002
          1006 1244 LEA 4, #0002, 4

アドレス オブジェクト ニモニック GR アドレス XR
コード フィールド フィールド フィールド
オペランド表示
```

 でカーソル（コロン（:）の消灯）を移動していけば、以降のアドレスの内容を見ることができます。また、 で戻すことができます。なお、 の代わりに  を使用しても同じ働きになります。


オブジェクトプログラムが記憶されていないときに、モニタ画面で  を押した場合はエラーになり“OBJECT ERROR”が表示されます。このときは **(CLS)** を押してメニュー画面に戻し、アセンブルを行ってください。


(注) オブジェクトプログラムは、CASLモードから他のモード(TEXT、RUN)へ切り替わったときに消去されます。他のモードからCASLモードに切り替えたときは、再度アセンブルを行ってください。

 **OUT命令について**

本機のマクロ命令のOUT a, bは、PUSH a、PUSH b、CALL 2とLEA 4, 2, 4の命令群を生成します。  
くわしくは271ページを参照してください。

## 6. 4 任意のアドレス内容の表示

メインプログラムの開始アドレスを表示している画面で、表示させたいアドレスを10進数、16進数、またはラベルで入力し、を押すと入力したアドレスの内容が表示されます。

#1008 

| << OBJECT >>  |      |     |       |
|---------------|------|-----|-------|
| ADDRESS=#1000 |      |     |       |
| 1008          | 6400 | JMP | #0004 |
| 1009          | 0000 |     |       |
| 100A          | 0000 | *   |       |
| 100B          | 00E8 |     |       |
| 100C          | 00E9 | *   |       |
| 100D          | 0043 |     |       |

**BREAK**を押せば、モニタ画面に戻ります。


(注) ● 入力したアドレスを命令語の第1語目とみなしますので、逆アセンブルの内容は正しく表示されない場合があります。

- 逆アセンブルした結果、ニモニックで表示する内容がない場合は \* マークを表示します。  
オブジェクトプログラムがないアドレスは、0000 \* 表示になります。
- 入力できるアドレスは#1000から、アセンブル終了時にGR4に入っているアドレス-1の範囲です。

## 6. 5 オブジェクトコードの書き換え

カーソル(コロン(:)の消灯)を書き換えたいアドレスへ移し、10進数、16進数、ラベル、文字でオブジェクトコード、またはそれに相当する内容を入力することにより、オブジェクトコードを書き換えることができます。

〈例〉書き換えたいアドレスの行を表示させ(:の消灯)

#1210 

のように操作します。

オブジェクトコードが書き換えられ、ニモニックも変わります。

また、アセンブラの命令語も入力できます。

- 10進数、16進数、ラベル、文字での入力の形式は255ページの「レジスタに値を設定する方法」を参照してください。

## 7. CASL実行例

次のプログラムは5個のデータの総和を求めるプログラムです。




データは、行番号130~170に定義されています。また、行番号120で総和を格納する領域を確保し、行番号90でその領域に総和を収納しています。

|     |      |       |               |                                   |
|-----|------|-------|---------------|-----------------------------------|
| 10  | EXAM | START |               |                                   |
| 20  | BGN  | LEA   | GR0, 0        | (GR0をクリアする)                       |
| 30  |      | LEA   | GR1, 0        | (GR1をクリアする)                       |
| 40  |      | JMP   | AGN1          | (無条件にAGN1へジャンプする)                 |
| 50  | AGN  | ADD   | GR0, DAT, GR1 | (指標レジスタGR1を利用してデータを順番にGR0へ加算する)   |
| 60  |      | LEA   | GR1, 1, GR1   | (GR1の内容に1を加えて、結果をGR1に入れる カウントアップ) |
| 70  | AGN1 | CPA   | GR1, N        | (GR1の内容とN番地の内容とを大小比較する個数の判断)      |
| 80  |      | JMI   | AGN           | (FRの値が“負”を示していればAGNへジャンプする)       |
| 90  |      | ST    | GR0, TTL      | (GR0の内容をTTL番地へ格納する)               |
| 100 |      | EXIT  |               | (プログラムの実行終了)                      |
| 110 | N    | DC    | 5             |                                   |
| 120 | TTL  | DS    | 1             | (結果を格納するためのエリアを確保する)              |
| 130 | DAT  | DC    | #000C         | } サンプルデータ                         |
| 140 |      | DC    | #07F3         |                                   |
| 150 |      | DC    | #0231         |                                   |
| 160 |      | DC    | #0009         |                                   |
| 170 |      | DC    | #000F         |                                   |
| 180 |      | END   |               | (プログラム終了)                         |

このプログラムをTEXTモードで入力し、CASLモードでアセンブルしてください。

## 7. 1 オブジェクトコードの内容確認

以下の手順に従って、モニタ機能によりオブジェクトコードの内容を確認します。

| キー操作                                                                                  | 表示                                                                                                             |
|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
|                                                                                       | *** CASL ***<br>Assemble Monitor Go                                                                            |
|  | << MONITOR >><br>Register Object                                                                               |
|  | << OBJECT >><br>ADDRESS=#1000                                                                                  |
|  | 1000 1200 LEA 0, #0000<br>1001:0000<br>1002:1210 LEA 1, #0000<br>1003:0000<br>1004:6400 JMP #100A<br>1005:100A |

|             |                                                                                                                                                                                                                                                                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ▼<br>:      | 1006:2001 ADD 0, #1014, 1<br>1007:1014<br>1008:1211 LEA 1, #0001, 1<br>1009:0001<br>100A:4010 CPA 1, #1012<br>100B:1012<br>100C:6100 JMI #1006<br>100D:1006<br>100E:1100 ST 0, #1013<br>100F:1013<br>1010:6400 JMP #0004<br>1011:0004<br>1012:0005 *<br>1013:0000<br>1014:000C *<br>1015:07F3<br>1016:0231 *****<br>1017:0009<br>1018:000F *<br>1019:0000 |
| BREAK<br>ON | << MONITOR >><br>Register Object                                                                                                                                                                                                                                                                                                                          |

## 7.2 ノーマル実行

次にソースプログラムの行番号80 JMI AGN にブレークポイントを設定して、プログラムの実行結果を調べることにします。

前項の方法でオブジェクトコードを調べると、この命令はアドレス#100Cに格納されていることがわかります。

| キー操作        | 表示                                                                                                 | 説明                                              |
|-------------|----------------------------------------------------------------------------------------------------|-------------------------------------------------|
|             | *** CASL ***<br>Assemble Monitor Go                                                                |                                                 |
| (M)         | << MONITOR >><br>Register Object                                                                   | モニタ機能を選びます。                                     |
| (R)         | GR0: #0000 0<br>GR1: #0000 0<br>GR2: #0000 0<br>GR3: #0000 0<br>GR4: #1AA5 6821<br>PC : #1000 4096 | レジスタの内容を呼び出し、カーソル(コロン(:)の消灯)をブレークポイントレジスタへ移します。 |
| ▼<br>:<br>▼ | BP #FFFF 65535                                                                                     | ブレークポイントは設定されていません。                             |
| #100C       | BP #FFFF #100C_                                                                                    | ブレークポイントをアドレス#100Cに設定します。                       |
| ↵           | BP #100C 4108                                                                                      |                                                 |

|                 |                                                                                                       |                                                                      |
|-----------------|-------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| ▼               | BC #0000 0                                                                                            | ブレークカウンタを2に設定します。(2回目の実行で停止)                                         |
| 2 ↵             | BC #0002 2                                                                                            |                                                                      |
| (BREAK) (BREAK) | *** CASL ***<br>Assemble Monitor Go                                                                   | メニュー画面に戻します。                                                         |
| (G) ↵           | << SIMULATION >><br>START ADDRESS=#1000<br>Normal Trace                                               | 実行方法選択画面にします。                                                        |
| (N)             | 100C: GR0:000C GR4:1AA5<br>* * GR1:0001 PC :100C<br>*STP* GR2:0000 FR :0002<br>* * GR3:0000 <JMI>     | ノーマル実行をします。2回目実行の#100C番地でブレークがかかり、レジスタの内容を表示します。GR0にデータ#000Cが加えられます。 |
| ↵               | 100C: GR0:07FF GR4:1AA5<br>* * GR1:0002 PC :100C<br>*STP* GR2:0000 FR :0002<br>* * GR3:0000 <JMI>     | 実行が再開され、次のデータ(#07F3)をGR0に加え、GR1は2になります。                              |
| (BREAK) (BREAK) | *** CASL ***<br>Assemble Monitor Go                                                                   |                                                                      |
| (M) (R)         | GR0: #07FF 2047<br>GR1: #0002 2<br>GR2: #0000 0<br>GR3: #0000 0<br>GR4: #1AA5 6821<br>PC : #100C 4108 | モニタでレジスタを呼び出します。                                                     |
| ▼<br>:<br>▼     | BC #0000 0                                                                                            | カーソルをブレークカウンタへ移します。                                                  |
| 4 ↵             | BC #0004 4                                                                                            | 4を設定します。                                                             |
| (BREAK) (BREAK) | *** CASL ***<br>Assemble Monitor Go                                                                   | メニュー画面に戻します。                                                         |
| (G) ↵           | << SIMULATION >><br>START ADDRESS=#100C<br>Normal Trace                                               | 実行方法選択画面にします。実行開始アドレスは先ほど、停止したアドレスになっています。                           |
| (N)             | 100C: GR0:0A48 GR4:1AA5<br>* * GR1:0005 PC :100C<br>*STP* GR2:0000 FR :0001<br>* * GR3:0000 <JMI>     | ノーマル実行をします。先に停止した状態からの継続実行になります。GR0には5個のデータの合計が入っています。               |
| (BREAK) (BREAK) | *** CASL ***<br>Assemble Monitor Go                                                                   | メニュー画面に戻します。                                                         |

(注) ● プログラムカウンタ (PC) とブレークポイント (BP) の値が一致したときに、ブレークカウンタ (BC) の値が0または1であればプログラムの実行を停止し、ブレークカウンタの値を0にします。ブレークカウンタの値が0または1以外のときは、ブレークカウンタから1を減じて、プログラムを続行します。

#### BC=2のときの実行状況

| 実行行番号 | 20行 → 30行 → 40行 → 70行 → 80行 → 50行 → 60行 → 70行 → 80行 → |
|-------|-------------------------------------------------------|
| BC=2  | 2 → 1 1 → 0                                           |
| GR0=0 | 0 → 000C 000C                                         |
|       | ↑<br>停止                                               |

● スタートアドレスとブレークポイント (BP) の値が等しいとき、ブレークカウンタ (BC) の値が0であればプログラムを続行し、次にプログラムカウンタとブレークカウンタの値が一致したときに停止します。  
また、ブレークカウンタの値が1であれば、何も実行せずに停止し、ブレークカウンタの値を0にします。ブレークカウンタの値が0および1以外のときは、1を減算してプログラムを続行します。

### 7. 3 ブレークポイントの解除

設定したブレークポイントの解除には、次の2通りの方法があります。

- ① アセンブルを再実行します。
- ② ブレークポイントの設定時と同じように、メニュー画面より  
 ...と操作して、BPレジスタを表示させます。次に-1 と操作すると解除できます。  
 でメニュー画面に戻します。

### 7. 4 トレース実行

同じプログラムをトレース実行してみましょう。

ブレークポイントは解除してください。なお次の例はアセンブル実行直後の例を示しています。

| キー操作 | 表 示                                                                                | 説 明                                        |
|------|------------------------------------------------------------------------------------|--------------------------------------------|
|      | *** CASL ***<br>Assemble Monitor Go                                                |                                            |
|      | << SIMULATION >><br>START ADDRESS=#1000<br>Normal Trace                            |                                            |
|      | 1000:GR0:0000 GR4:1AA5<br>GR1:0000 PC :1002<br>GR2:0000 FR :0001<br>GR3:0000 <LEA> | トレースモードで実行を開始します。<br>LEA 0, 0でGR0をリセットします。 |
|      | 1002:GR0:0000 GR4:1AA5<br>GR1:0000 PC :1004<br>GR2:0000 FR :0001<br>GR3:0000 <LEA> | LEA 1, 0でGR1をリセットします。                      |

|  |                                                                                    |                                                       |
|--|------------------------------------------------------------------------------------|-------------------------------------------------------|
|  | 1004:GR0:0000 GR4:1AA5<br>GR1:0000 PC :100A<br>GR2:0000 FR :0001<br>GR3:0000 <JMP> | JMP AGN1に無条件に分岐 (ジャンプ) します。PCの値を見れば100A番地であることがわかります。 |
|  | 100A:GR0:0000 GR4:1AA5<br>GR1:0000 PC :100C<br>GR2:0000 FR :0002<br>GR3:0000 <CPA> | CPA 1, NでGR1の内容とN番地の内容と比較します。                         |
|  | 100C:GR0:0000 GR4:1AA5<br>GR1:0000 PC :1006<br>GR2:0000 FR :0002<br>GR3:0000 <JMI> | JMI AGNでFRが2のときはAGNアドレスへ分岐します。                        |
|  | 1006:GR0:000C GR4:1AA5<br>GR1:0000 PC :1008<br>GR2:0000 FR :0000<br>GR3:0000 <ADD> | ADD 0, DAT, 1で指標レジスタ (GR1) を利用してデータをGR0に加算します。        |
|  | 1008:GR0:000C GR4:1AA5<br>GR1:0001 PC :100A<br>GR2:0000 FR :0000<br>GR3:0000 <LEA> | GR1に1を加算します。                                          |
|  | 100A:GR0:000C GR4:1AA5<br>GR1:0001 PC :100C<br>GR2:0000 FR :0002<br>GR3:0000 <CPA> | 以降、同様な動作がデータ数分繰り返されます。                                |
|  | 100A:GR0:0A48 GR4:1AA5<br>GR1:0005 PC :100C<br>GR2:0000 FR :0001<br>GR3:0000 <CPA> |                                                       |
|  | 100C:GR0:0A48 GR4:1AA5<br>GR1:0005 PC :100E<br>GR2:0000 FR :0001<br>GR3:0000 <JMI> | CPA 1, NでFRが1となったため、分岐せずに次の命令を実行します。                  |
|  | 100E:GR0:0A48 GR4:1AA5<br>GR1:0005 PC :1010<br>GR2:0000 FR :0001<br>GR3:0000 <ST>  | ST 0, TTLでGR0の値を番地TTLへ格納します。                          |
|  | 1010:GR0:0A48 GR4:1AA5<br>GR1:0005 PC :0004<br>GR2:0000 FR :0001<br>GR3:0000 <JMP> | EXIT命令により実行を停止します。(EXIT処理ルーチン#0004番地へ分岐)              |
|  | *** CASL ***<br>Assemble Monitor Go                                                | メニュー画面に戻ります。                                          |



## 7. 5 空欄穴埋め問題の実例

前の例題でプログラムの行番号60が空欄穴埋め問題となっていた場合、下の例のように60ラインに\*を入力します。\*命令は[\*]を押して入力します。

〈例〉

```

10EXAM START
20BGN LEA GR0,0
30 LEA GR1,0
40 JMP AGN1
50AGN ADD GR0,DAT,GR
1
60 LEA GR1,1,GR1
70AGN1 CPA GR1,N
80 JMI AGN
90 ST GR0,TTL
100 EXIT
110N DC 5
120TTL DS 1
130DAT DC #000C
140 DC #07F3
150 DC #0231
160 DC #0009
170 DC #000F
180 END

```



〈例〉

```

10EXAM START
20BGN LEA GR0,0
30 LEA GR1,0
40 JMP AGN1
50AGN ADD GR0,DAT,GR
1
60 *
70AGN1 CPA GR1,N
80 JMI AGN
90 ST GR0,TTL
100 EXIT
110N DC 5
120TTL DS 1
130DAT DC #000C
140 DC #07F3
150 DC #0231
160 DC #0009
170 DC #000F
180 END

```

上記、右のプログラムをアセンブルすると行番号60に対応するオブジェクトコードが  
“#0000 #0000”になります。

〈例〉


```

ADD : OBJECT : LINE NO.
      : 10
1000:1200 0000: 20
1002:1210 0000: 30
1004:6400 100A: 40
1006:2001 1014: 50
1008:0000 0000: 60
100A:4010 1012: 70
100C:6100 1006: 80
100E:1100 1013: 90
1010:6400 0004: 100
1012:0005 : 110
1013:0000 : 120
1014:000C : 130
1015:07F3 : 140
1016:0231 : 150
1017:0009 : 160
1018:000F : 170
      : 180

```

このプログラムを実行すると、#1008番地でプログラムの実行を停止しますので、レジスタの値を確認できます。また、モニタ機能を使って、オブジェクトの確認や修正ができます。

| キー操作                 | 表 示                                                                                                                | 説 明                           |
|----------------------|--------------------------------------------------------------------------------------------------------------------|-------------------------------|
|                      | *** CASL ***<br>Assemble Monitor Go                                                                                |                               |
|                      | << SIMULATION >><br>START ADDRESS=#1000<br>Normal Trace                                                            |                               |
|                      | 1008: GR0:000C GR4:1AAC<br>* * GR1:0000 PC :1008<br>*STP* GR2:0000 FR :0000<br>* * GR3:0000 <*>                    | *命令があるとプログラムの実行を停止します。        |
|                      | 1008: GR0:0018 GR4:1AAC<br>* * GR1:0000 PC :1008<br>*STP* GR2:0000 FR :0000<br>* * GR3:0000 <*>                    | (注1)<br>(注2)                  |
|                      | *** CASL ***<br>Assemble Monitor Go                                                                                |                               |
| <br>#1008            | << OBJECT >><br>ADDRESS=#1008_                                                                                     | アドレス#1008のオブジェクトプログラムを呼び出します。 |
|                      | 1008 0000 *<br>1009:0000<br>100A:4010 CPA 1, #1012<br>100B:1012<br>100C:6100 JMI #1006<br>100D:1006                |                               |
| LEA  GR<br>1, 1, GR1 | 1008 0000 LEA GR1, 1, GR1_<br>1009:0000<br>100A:4010 CPA 1, #1012<br>100B:1012<br>100C:6100 JMI #1006<br>100D:1006 | 空欄の答えを入力します。                  |
|                      | 1008 1211 LEA 1, #0001, 1<br>1009:0001<br>100A:4010 CPA 1, #1012<br>100B:1012<br>100C:6100 JMI #1006<br>100D:1006  | アセンブルし、オブジェクトが生成されます。         |
| <br>'BGN'            | << SIMULATION >><br>START ADDRESS='BGN'_                                                                           | もう一度初めからプログラムを実行します。          |
|                      | << SIMULATION >><br>START ADDRESS=#1000<br>Normal Trace                                                            | 終了するとメニュー画面に戻ります。             |
|                      | *** CASL ***<br>Assemble Monitor Go                                                                                |                               |
| <br>'TTL'            | 1013 0A48 *****<br>1014:000C<br>1015:07F3 *****<br>1016:0231<br>1017:0009 *<br>1018:000F                           | (注3)                          |

- (注1) シミュレーションで\*命令を実行し、プログラムが停止しているときに  を押すと、次の命令から実行を開始します。実行後、\*命令があると再び停止します。
- (注2) この時点でGR0には#07FF、GR1には#0001が入っているべきですが、値が異なっていることがわかります。
- (注3) 合計を確認すると正しい値が入っています。つまり、空欄に入れた答えが正しいことがわかります。この時点では、オブジェクトコードは変更されていますが、ソースプログラムは変更されていません。ソースプログラムはテキストモードに戻って変更します。

## 8. COMETの仕様

### 8. 1 仮想計算機COMETと本機CASLとの相異点

本機のCASLの仕様は、経済産業省の設定している仮想計算機COMETのCASLと比べて、以下の点が異なります。(平成13年度より、COMETはCOMET IIに、CASLはCASL IIに仕様改定されています。)


#### (1) START命令

ラベルが省略できます。また、オペランドが省略された場合、#1000番地からプログラムを実行します。

#### (2) DC命令

オペランドが10進数で-32768~65535の範囲にないときは、アセンブルするとエラーになります。

#### (3) IN命令 (CALL #0000)

IN命令を実行すると、画面に“?”が表示され、キーからの入力が可能になります。このとき、 の入力があると、入力文字長に-1 (#FFFF)を設定します。


この機能は、入力装置としてカードリーダーを想定したときに利用すると便利です。

#### (4) OUT命令 (CALL #0002)

OUT命令を実行すると、“PRINT”が点灯しているときは印字し、点灯していないときは画面に表示します。

出力文字数が97文字以上のときはエラーになります。

#### (5) WRITE命令 (CALL #0006)

WRITE命令を実行するとレジスタの内容を表示し、プログラムの実行を停止します。このとき  を押せばプログラムの実行を続行します。

#### (6) 複数プログラムの連結

本機には、別々にアセンブルして作成したオブジェクトプログラムを連結する“リンカ”の機能はありません。このような動作をさせたいときは、個々にSTART命令とEND命令で囲んだ複数のプログラムを一度にアセンブルしてください。なお、アセンブルするとき、他のプログラムのラベルと同じものがあるとエラーになります。

### 8. 2 COMETの仕様概略

アセンブラ言語の理解を深めるため、本機のCASLモードでの仮想ハードウェア仕様 (COMETに準拠) について概略を説明します。

(1) 1語長 : 16ビット

(2) 制御方式 : 逐次制御方式 (ノイマン型)

(3) 数値の表現 : 16ビットの2進数、負数は2の補数表示

(4) レジスタ : ①GR0~4 (16ビット) General Register 汎用レジスタ

GR1~GR4は指標レジスタとしても使用。さらにGR4はスタックポインタとしても使用される。

②PC (16ビット) Program Counter プログラムカウンタ

実行中の命令語の先頭アドレスを保持するレジスタ。


③FR (2ビット) Flag Register フラグレジスタ

演算、比較結果の情報保持レジスタ。

| GRに設定されたデータ |    |    |    |
|-------------|----|----|----|
|             | 正  | ゼロ | 負  |
| FRの値        | 00 | 01 | 10 |

(2進表記)

[補足] トレース実行を行うとレジスタの内容を容易に確認できます。

 **なぜ01がゼロなのか**

LEA命令でGRに設定される値がゼロのとき、FRには01が設定されます。そのため、FRの01をゼロと呼び、JZE (Jump on ZERo) で分岐します。分岐命令においては注意してください。

### 8. 3 命令語の構成

CASLの命令は2語長と定義されています。その構成をまとめます。

| 第1語 |    | 第2語 |     | 命令語とアセンブラとの対応 |             |                        |
|-----|----|-----|-----|---------------|-------------|------------------------|
| OP  |    |     |     | 書 き か た       |             | ニモニックスペル               |
| 主 副 | GR | XR  | adr | 命令コード         | オペランド       |                        |
| 0 0 |    |     |     | 未使用           |             |                        |
| 1 0 |    |     |     | LD            | GR, adr, XR | load                   |
| 1 1 |    |     |     | ST            | GR, adr, XR | store                  |
| 1 2 |    |     |     | LEA           | GR, adr, XR | load effective address |
| 2 0 |    |     |     | ADD           | GR, adr, XR | add arithmetic         |
| 2 1 |    |     |     | SUB           | GR, adr, XR | subtract arithmetic    |
| 3 0 |    |     |     | AND           | GR, adr, XR | and                    |
| 3 1 |    |     |     | OR            | GR, adr, XR | or                     |
| 3 2 |    |     |     | EOR           | GR, adr, XR | exclusive or           |

|             |   |   |         |                 |                        |
|-------------|---|---|---------|-----------------|------------------------|
| 4 0         |   |   |         | CPA GR, adr, XR | compare arithmetic     |
| 4 1         |   |   |         | CPL GR, adr, XR | compare logical        |
| 5 0         |   |   |         | SLA GR, adr, XR | shift left arithmetic  |
| 5 1         |   |   |         | SRA GR, adr, XR | shift right arithmetic |
| 5 2         |   |   |         | SLL GR, adr, XR | shift left logical     |
| 5 3         |   |   |         | SRL GR, adr, XR | shift right logical    |
| 6 0         | 0 |   |         | JPZ adr, XR     | jump on plus or zero   |
| 6 1         | 0 |   |         | JMI adr, XR     | jump on minus          |
| 6 2         | 0 |   |         | JNZ adr, XR     | jump on non zero       |
| 6 3         | 0 |   |         | JZE adr, XR     | jump on zero           |
| 6 4         | 0 |   |         | JMP adr, XR     | unconditional jump     |
| 7 0         | 0 |   |         | PUSH adr, XR    | push effective address |
| 7 1         |   | 0 | 0 0 0 0 | POP GR          | pop up                 |
| 8 0         | 0 |   |         | CALL adr, XR    | call subroutine        |
| 8 1         | 0 | 0 | 0 0 0 0 | RET             | return from subroutine |
| 9<br>7<br>F |   |   |         | 未使用             |                        |

## 8. 4 命令の種類と機能

本機はCASLで定義された23の命令を持っています。以下に各命令の機能を説明します。なお、説明には次の表記法を使います。

- ①GR : GRの値を番号とする汎用レジスタを示します。(ただし、 $0 \leq GR \leq 4$ の範囲)
- ②XR : XRの値を番号とする指標レジスタを示します。(ただし、 $1 \leq XR \leq 4$ の範囲)  
指標レジスタとしてGR1～GR4が使用できます。
- ③SP : スタックポインタを示します。(スタックポインタはGR4を使用)
- ④adr : ラベル名に対応する番地、または10進の定数(ただし、 $-32768 \leq \text{adr} \leq 65535$ )を示します。adrはアドレスとして0～65535の値を持ちますが、32768～65535の値を負の10進定数で記述することもできます。(例: 65535番地は-1と記述できます。)
- ⑤有効アドレス: adrとXRの内容とのアドレス加算値、またはその値が示す番地です。
- ⑥(X) : X番地の内容、またはXが汎用レジスタ(GR)を示す場合は、そのGRの内容を表します。
- ⑦[ ] : [ ]に囲まれた部分は省略可能です。XRを省略した場合は、指標レジスタによる修飾は行われません。

### 【各命令と機能】

以下の説明においてAとはA番地に格納されている内容を示します。GR0～GR4においてもそれぞれのレジスタに格納されている内容(値)を示します。

- (1)LD GR, adr [, XR]  
機能: 有効アドレスの内容をGRにロードします。

- (2)ST GR, adr [, XR]  
機能: 有効アドレスにGRの内容を格納します。
- (3)LEA GR, adr [, XR]  
機能: 有効アドレスそのものをGRにロードします。GRの値によりFRが設定されます。  
〈例〉LEA GR1, 100 定数100をGR1に格納します。  
LEA GR1, 10, GR1 GR1に10をたすこととなります。  
LEA GR1, 0, GR2 GR2の内容をGR1に移動させます。
- (4)ADD GR, adr [, XR]  
機能: 有効アドレスの内容をGRに加算します。GRの値によりFRが設定されます。
- (5)SUB GR, adr [, XR]  
機能: 有効アドレスの内容をGRから減算します。GRの値によりFRが設定されます。

〈例〉右図の内容のときは次のようになります。ただし、A番地には(1010)<sub>16</sub>が格納されているものとします。したがって、A+GR2は(1015)<sub>16</sub>になります。

| (実行前の状態)        |                                    |     |                         |
|-----------------|------------------------------------|-----|-------------------------|
| LD GR1, A, GR2  | (1015) <sub>16</sub> 番地の30をGR1にロー  | GR1 | 15                      |
| ド               |                                    |     |                         |
| ST GR1, A, GR2  | GR1の15を(1015) <sub>16</sub> 番地に格納  | GR2 | 5                       |
| LEA GR1, A, GR2 | (1015) <sub>16</sub> をGR1に格納       |     | 20 (1010) <sub>16</sub> |
| ADD GR1, A, GR2 | (1015) <sub>16</sub> 番地の30をGR1に加算  |     | 番地                      |
| SUB GR1, A, GR2 | (1015) <sub>16</sub> 番地の30をGR1から減算 |     | 30 (1015) <sub>16</sub> |
| 番地              |                                    |     |                         |

- (6)AND GR, adr [, XR]
- (7)OR GR, adr [, XR]
- (8)EOR GR, adr [, XR]

機能: 1語16ビットのビットごとの論理積(AND)、論理和(OR)、排他的論理和(EOR)をGRの内容と有効アドレスの内容とで行い、その結果をGRに設定します。GRの値によりFRが設定されます。

〈例〉AND GR1, MASK MASK番地の内容を7にしておくと、GR1の下位3ビットが取り出せます。  
EOR GR1, MASK MASK番地の内容をFFFF(-1)にしておくと、GR1の全ビットが反転します。

- (9)CPA GR, adr [, XR]
- (10)CPL GR, adr [, XR]

機能: GRの内容と有効アドレスの内容との算術比較(CPA)または論理比較(CPL)を行います。比較結果によりFRに次の値を設定します。

| 比較結果            | FRの設定ビット値 |
|-----------------|-----------|
| (GR) > (有効アドレス) | 00 (0) 正  |
| (GR) = (有効アドレス) | 01 (1) ゼロ |
| (GR) < (有効アドレス) | 10 (2) 負  |

( )はGRまたは有効アドレスの内容を示します。

算術比較: 符号付の数として比較  
論理比較: 16ビットの整数として比較

〈例〉GR 1の内容が $(-256)_{10}$ でA番地の内容が $(256)_{10}$ のときは

CPA GR 1, A でFRは $(10)_2$ になります。

CPL GR 1, A でFRは $(00)_2$ になります。

16ビット表記では $(-256)_{10}$ は補数表示され $(FF00)_{16}$ になり、 $(256)_{10}$ は $(0100)_{16}$ になり、 $(FF00)_{16} > (0100)_{16}$ です。

(11)JPZ adr [, XR]

(12)JMI adr [, XR]

(13)JNZ adr [, XR]

(14)JZE adr [, XR]

機能：FRの値により有効アドレスに分岐(ジャンプ)します。分岐しないときは次の命令に移ります。

| 命令  | 分岐するときのFRの値 | 意 味        |
|-----|-------------|------------|
| JPZ | 00 または 01   | 正または0のとき分岐 |
| JMI | 10          | 負のとき分岐     |
| JNZ | 00 または 10   | 正または負のとき分岐 |
| JZE | 01          | 0のとき分岐     |

(15)JMP adr [, XR]

機能：FRの値に関係なく無条件に有効アドレスに分岐(ジャンプ)します。



#### FRレジスタについて

分岐命令はFRの値で分岐先を決めますが、FRの値を決定する命令は次の命令です。

LEA、ADD、SUB、AND、OR、EOR、CPA、CPL、SLA、SRA、SLL、SRL

(16)SLA GR, adr [, XR]

(17)SRA GR, adr [, XR]

機能：符号ビット(最上位ビット)を除いたGRの内容を、有効アドレスで指定されたビット数だけ左(SLA)または右(SRA)ヘシフトします。シフトしてはみ出たビットは切り捨てられます。空いたビットには左シフトのときは0が入り、右シフトのときは符号ビットが入ります。シフトした後のGRの内容によりFRが設定されます。GRが正のときはFRは $(00)_2$ になります。

(18)SLL GR, adr [, XR]

(19)SRL GR, adr [, XR]

機能：符号ビットを含んだGRの内容を、有効アドレスで指定されたビット数だけ左(SLL)または右(SRL)ヘシフトします。シフトしてはみ出たビットは切り捨てられます。空いたビットには0が入ります。シフトした後のGRの内容によりFRが設定されます。GRが正のときはFRは $(00)_2$ になります。

〈例〉GR 1=1001 1100 0100 1011でGR 2=3のとき

SLA GR 1, 3 \*3ビット左ヘシフトし GR 1=1110 0010 0101 1000 になります。

(算術左シフト)

SRA GR 1, 3 \*3ビット右ヘシフトし GR 1=1111 0011 1000 1001 になります。

(算術右シフト)

SLL GR 1, 1, GR 2 \*4ビット左ヘシフトし GR 1=1100 0100 1011 0000 になります。

(論理左シフト)

SRL GR 1, 1, GR 2 \*4ビット右ヘシフトし GR 1=0000 1001 1100 0100 になります。

(論理右シフト)

(20)PUSH adr [, XR]

機能：SP(スタックポインタ)から1をアドレス減算した後、SPで指定された番地に有効アドレスを格納します。

(21)POP GR

機能：SPで指定された番地の内容をGRに設定してから、SPに1をアドレス加算します。

(22)CALL adr [, XR]

機能：有効アドレス(サブルーチン)に分岐する命令で、分岐する前にスタックに、戻り番地(次の命令の番地)を保持します。

(23)RET

機能：CALLに対する戻り命令で、スタックに保持されていた戻り番地により、メインプログラムに移ります。

## 8. 5 アセンブラの文法

本機のアセンブラ言語はCASLと同様に、疑似命令、マクロ命令、機械語命令を持っています。疑似命令、マクロ命令、機械語命令は次のように記述します。

| ラベル欄  | 命令コード | オペランド欄      | 注釈欄  |
|-------|-------|-------------|------|
| [ラベル] | START | [実行開始番地]    | [注釈] |
| 空白    | END   | 空白          | [注釈] |
| [ラベル] | DC    | 定数          | [注釈] |
| [ラベル] | DS    | 領域の語数       | [注釈] |
| [ラベル] | IN    | 入力領域、入力文字長  | [注釈] |
| [ラベル] | OUT   | 出力領域、出力文字長  | [注釈] |
| [ラベル] | EXIT  | 空白          | [注釈] |
| [ラベル] | WRITE | 空白          | [注釈] |
| [ラベル] | 機械語命令 | 命令の種類と機能 参照 | [注釈] |

表中の「空白」は記入してはならない箇所、[ ] で囲まれた部分は省略可能を意味します。

## 8. 6 疑似命令

疑似命令はアセンブラの制御、定数の定義、プログラムの連結のために必要なデータの生成などを行うもので、次の4種類があります。

(1)START：実行開始番地の定義や、他のプログラムとの連結のための入りの名前の定義をします。プログラムの最初に必要です。

(2)END：プログラムの終わりを定義します。プログラムの最後に必要です。



(3)DC : define constant の略で、定数で指定したデータを格納します。  
定数には次の4種があります。

- ①10進定数 DC n  
nで指定した10進数値を1語の2進数データとして格納します。nは-32768～65535の範囲です。
- ②16進定数 DC #h  
hは4桁の16進数(0～F)です。hで指定された16進数値を1語の2進数データとして格納します。  
#hは#0000～#FFFFの範囲です。
- ③文字定数 DC '文字列'  
文字列の左端から1文字ずつ連続する語の下位8ビット(第8～15ビット)に文字データを格納します。  
各語の第0～7ビットには0のビットが入ります。文字列には384ページのキャラクタ・コード表の内、コード32～38(&H20～&H26)、40～95(&H28～&H5F)、97～122(&H61～&H7A)および166～223(&HA6～&HDF)の文字が使えます。なお、文字列の長さは0であってはなりません。
- ④アドレス定数 DC ラベル名  
ラベル名に対応するアドレス値を1語の2進数データとして格納します。

(4)DS : define storage の略で、指定した語数の領域を確保します。  
領域の語数は、10進定数(≥0)で指定します。0の場合、領域は確保されずラベル名だけが有効となります。

8. 7 マクロ命令

- (1)IN 入力領域, 入力文字長:  
キーボードから1レコードの文字データを入力領域のラベル名の番地に格納します。入力文字長のラベル名の番地には、入力した文字数が格納されます。
- (2)OUT 出力領域, 出力文字長:  
出力領域に格納されている文字データを1レコードとして、表示またはプリンタに出力します。ただし、出力文字長のラベル名の番地で指定されている文字数分だけ出力します。
- (3)EXIT : プログラムの実行を終了します。
- (4)WRITE : レジスタGR0～GR4およびPC、FRの内容を表示します。また、プリンタが印字できる状態になっているときは、GR0～GR3の内容を印字します。

| 〈例〉  |               | 説 明                            |  |
|------|---------------|--------------------------------|--|
| 10   | STAR1         |                                |  |
| 20   | IN A,C        | キーボードで文字を入力                    |  |
| 25   | OUT NL,N      | 入力表示と出力表示の区別のため、強制的に表示内容を変えている |  |
| 30   | OUT A,B       | キーボードで入力した文字の出力                |  |
| 40   | EXIT          |                                |  |
| 50A  | DS 20         | 入力文字数を20に設定                    |  |
| 60B  | DC 2          | 出力文字数を2に設定                     |  |
| 70C  | DS 1          | 入力文字長を格納するアドレス                 |  |
| 80N  | DC 9          |                                |  |
| 90NL | DC 'PPPPPPPPP | } 強制表示内容の指定                    |  |
| ,    |               |                                |  |
| 100  | END           |                                |  |

[キー操作]  
(アセンブル実行後)  
G 開始アドレスは#1000  
N  
QWERTYUIOP P P P P P P P P P (強制表示)  
 QW  
 メニュー表示

- ①行番号30をOUT A, Cに変えて実行してみてください。
  - ②行番号25を削除して実行してみてください。
  - ③行番号60の「2」を変えて実行してみてください。
- いろいろ変えて実行することにより、その違いがわかります。

本機のマクロ命令では、次の命令群を生成します。

|       |      | <オブジェクト> |         | <命令>        |
|-------|------|----------|---------|-------------|
| IN    | a, b | 7 0 0 0  | a a a a | PUSH a      |
|       |      | 7 0 0 0  | b b b b | PUSH b      |
|       |      | 8 0 0 0  | 0 0 0 0 | CALL 0      |
|       |      | 1 2 4 4  | 0 0 0 2 | LEA 4, 2, 4 |
| OUT   | a, b | 7 0 0 0  | a a a a | PUSH a      |
|       |      | 7 0 0 0  | b b b b | PUSH b      |
|       |      | 8 0 0 0  | 0 0 0 2 | CALL 2      |
|       |      | 1 2 4 4  | 0 0 0 2 | LEA 4, 2, 4 |
| EXIT  |      | 6 4 0 0  | 0 0 0 4 | JMP 4       |
| WRITE |      | 8 0 0 0  | 0 0 0 6 | CALL 6      |

(注) a a a a、b b b bはラベルaまたはラベルbが定義されているアドレスを示します。

8. 8 特殊命令

- (1)   
機能：情報処理技術者試験の空欄穴埋め問題に対応するための、空欄入力命令です。  
この命令をアセンブルすると#00000が2語生成されます。シミュレーションのときには命令語の第1語が#00XX(XXは何でもかまいません)の命令を実行後一時停止し、\*STP\*と表示します。このときオブジェクトの変更やレジスタ値の確認をすることができます。その後、プログラムを続行できます。

# 第 8 章

## 機械語モニタとアセンブラ機能

プログラム言語は用途により色々な種類があります。本機では、高級言語である BASIC や機械語も使用できます。

BASIC は英語に近い形で使いやすい言語ですが、処理スピードの点では機械語に劣ります。一方、機械語は人間にとって非常にわかりにくい言語ですが、いろいろなことができ、コンピュータの機能を最大限に活用できます。

本機はこの機械語を使用するために、機械語モニタとアセンブラ機能が内蔵されています。

機械語モニタ機能では、メモリ内の機械語の表示、メモリ内の書き換え、機械語の実行などが簡単な命令だけで行えます。

アセンブラ機能は、ニモニック（アセンブラ言語）で書かれたソースプログラムを機械語に変換する機能です。

このポケットコンピュータは CPU（シーピーユー：中央演算処理装置）に、優れた 8 ビット CPU として広く利用されている Z80（CMOS Z80 相当品）を使用しています。

Z80 は人気のある CPU であり、機械語やニモニックについての入門書等の関連書籍が市販されていますので、それらの書籍を参照してください。

この章では、まず機械語モニタ機能の各命令（コマンド）の働きについて説明し、その後ソースプログラムの作成、アセンブルのしかたなどを説明します。

### 機械語を使う場合のご注意

1. 機械語を実行したときに、プログラムや操作に誤りがあると、次のようなことが起こります。

① プログラムを実行し続けて、すべてのキーが動かなくなる。

この場合は、リセットスイッチを押します。

② でたらめな表示や動作を続ける。

この場合は、**[BREAK]** を押します。それでも止まらないときは、リセットスイッチを押します。

③ プログラムやデータの一部またはすべてが、壊れたり、消えたりする。

これは機械語プログラムだけでなく、他のプログラム、ファイル、データに及ぶことがあります。

これらが起こったときは、計算機の内部の状態がどのようなになっているかわかりませんので、リセットスイッチを押し、すべての内容を消去してください。

#### ご注意

機械語プログラムを扱うときは、上記のようなことが起こる可能性があります。消えては困るプログラムやデータは、紙に書き写しておいてください。プリンタをお持ちの場合は、印字しておいてください。

2. 確保した機械語エリアだけを使用してください。

機械語モニタの USER コマンドで確保した機械語エリア以外を使用すると、BASIC や TEXT のプログラムなど、ほかの記憶内容を壊したり、正しい動作をしないことがあります。

### 参 考

機械語関係にはむずかしい用語がありますので、主な用語をまとめておきます。

**機械語** コンピュータが直接、解読・実行できる言語です。16 進法で表されます（コンピュータ内では 2 進数で扱われます）。

**ニモニック** 機械語コードを人間が記憶しやすいように符号化したものです。  
たとえば加算命令（addition）のコードを ADD のように定めます。  
ニモニックを言語的に体系付けたものをアセンブラ（アセンブリ）言語と呼びます。

#### ソースプログラム

ニモニック（アセンブラ言語）で書かれたプログラム。機械語プログラムに変換する元になるプログラムです。

#### オブジェクトプログラム

変換して得られたプログラムで、そのまま計算機で実行可能な形にされたものです。一般的には、ソースプログラムをアセンブルして得られた機械語プログラムのことです。単にオブジェクトと呼ぶこともあります。（オブジェクトは、機械語のコード 1 つ 1 つを指す場合と機械語プログラム全体を指す場合があります。）

**アセンブル** ソースプログラムを機械語プログラムに変換（翻訳）することです。  
人間が人手によりアセンブルを行うことを**ハンドアセンブル**と呼びます。


**アセンブラ** アセンブルをコンピュータに行わせるための翻訳プログラムのことです。  
**疑似命令** オブジェクトを格納する場所（アドレス）を指定したり、データを生成させたりするアセンブラ制御用の命令です。それ自体は機械語に変換されません。

**生成** 作り出すこと。ニモニックからオブジェクトを作り出すことを言います。

**アドレス** メモリにつけられている番号をアドレス（番地）と呼びます。  
機械語では、メモリを使用するときアドレスを指定して使用します。

## 機械語モニタ機能

### 1. 機械語モニタを使ううえでのきまり

機械語モニタは機械語モニタモードで使います。BASIC モード（RUN または PRO モード）で MON  と押せば、機械語モニタモードに切り替わります。


右の画面が表示されます。

（パスワードが設定されているときは、機械語モニタモードにできません。）

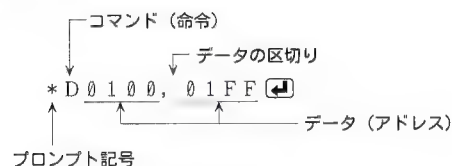
MACHINE LANGUAGE MONITOR

\*

\*記号は機械語モニタモードのプロンプト記号です。

コマンド（命令）は必ずこのプロンプト記号の後に書き、アドレス（番地）やデータの指定が必要な場合は、コマンドに続けて指定します。そして  を押して、実行させます。

〈例〉



- アドレスおよびデータの指定は16進数で行います。
- アドレスおよびデータの区切りはコンマ（,）で指定します。
- アドレスおよびデータの指定に0～Fの16進数字とコンマ以外を入れるとエラー（SYNTAX ERROR）になります。
- 機械語モニタモードは、**BASIC**、**TEXT** の操作または電源を切ること解除されます。

## 2. 機械語モニタの各命令の説明

### \*USER……ユーザーエリア

**機 能** 機械語エリアの確保および確保したエリアの表示を行います。

**書 式** (1) USER 終了番地   
 (2) USER   
 (3) USER 00FF 

**説 明** ● 書式(1)を実行するとメモリの0100H番地（開始番地）から指定した終了番地までを機械語エリアとして確保します。開始番地は自動的に0100H番地になります。

USER 01FF  と実行した例

```
*USER01FF
FREE:0100-01FF
*
```


↑ 確保した範囲

- 書式(2)を実行すると、現在確保されている機械語エリアの番地を表示します。

```
*USER
FREE:0100-01FF
*
```

機械語エリアが確保されていないときは“FREE:NOT RESERVED”と表示されます。

- 書式(3)を実行すると、機械語エリアを消去します。そして、次の表示をします。


USER 00FF 

“FREE:NOT RESERVED”

- 確保できない範囲（機械語エリアとして使用できない範囲）を指定するとエラー（MEMORY ERROR）になります。

### \*S………セットメモリ






**機 能** メモリの内容を書き換えます。

**書 式** (1) S 開始番地   
 (2) S 

**説 明** ● 書式(1)を実行すると、指定した開始番地の現在のメモリ内容が表示され、変更するデータの入力待ちになります。

S 0100  と実行した例

```
*S0100
0100:10-
↑
現在のメモリ内容
```

- データを変更するときは、1バイトデータ（2桁の16進の数字）を入力し、 を押します。データが変更され、次のアドレス（番地）のデータ入力待ちになります。データを変更しない場合は、データを入力せずに  のみを押します。そのまま次のアドレスのデータ入力待ちになります。
- データは2桁以上入力できません。入力途中の値を取り消すときは  または **CLS** を押します。
-  で、次のアドレスを呼び出すことができます。また  で前のアドレスを呼び出すことができます。
- 書式(2)を実行すると、以前のSコマンドで表示していた次のアドレスのメモリ内容が呼び出されます。電源を入れた後（初期値）では0100の指定になります。
- **BREAK** を押せばコマンド（命令）待ちの状態に戻ります。

〈例〉 次のオブジェクトプログラムを0100H番地から書き込みます。

〈オブジェクトプログラム〉

```
3E 01
18 04
3A 0F 01
3C
32 0F 01
C9
```

＝参考＝ 上記オブジェクトプログラムのソースプログラムを示します。

〈ソースプログラム〉

|                   |                       |
|-------------------|-----------------------|
| LD A, 01H         | ←Aレジスタに01Hを入れる        |
| JR ST             | ←ラベルSTの番地へジャンプ        |
| LD A, (010FH)     | ←Aレジスタに010FH番地の内容を入れる |
| INC A             | ←Aレジスタの内容に1を加算する      |
| ST: LD (010FH), A | ←Aレジスタの内容を010FH番地へ入れる |
| RET               | ←リターン                 |

[キー操作]

[表 示]

[BREAK]

USER 01FF

\*USER 01FF

FREE: 0100-01FF

{ 0100H番地から01FFH  
番地までを機械語エリアとして  
確保 (余裕のある大きさです)

S 0100

\*S 0100

3E

0100: 10-3E

01

0101: 12-01

18

0102: 00-18

04

0103: 00-04

3A

0104: 01-3A

0F

0105: 10-0F

01

0106: 50-01

3C

0107: 43-3C

32

0108: 2D-32

0F

0109: 47-0F

01

010A: 38-01

C9

010B: 30-C9

[BREAK]

010C: 31-

\*      }

ここには以前が表示されるため、このとおりの数値になるとは限りません。

## \* D……………ダンプメモリ

**機 能** メモリの内容を表示させます。

**書 式** (1) D 開始番地

(2) D

(3) D 開始番地, 終了番地

**説 明** ● 書式(1)を実行すると、開始番地から24バイト分 (6行分) のメモリの内容が表示されます。  
(プリントモードでは、開始番地から16バイト分 (4行分) を印字します。)

D 0100 と実行した例

〈表示例〉

16バイト単位の先頭アドレス→  
チェックサム→

|      |   |    |    |    |    |       |
|------|---|----|----|----|----|-------|
| 0100 | : | 3E | 01 | 18 | 04 | >...  |
| (1D) |   | 3A | 0F | 01 | 3C | :...< |
|      |   | 32 | 0F | 01 | C9 | 2.../ |
|      |   | 31 | 00 | 00 | 00 | 1...  |

010C番地以降は前に記憶  
していた内容が残っています。  
そのためチェックサムもこの  
ようになるとは限りません。

各データをアスキーコードとした  
場合の該当文字を表示します。た  
だし、00H~1FH、F9H~  
FFHのデータはピリオド(.)  
で表示されます。

● 開始番地、終了番地のアドレス (番地) の区分は規定 (XXXH~XXXFH番地) されて  
いて、その区分内であればどのアドレスを指定しても同じになります。たとえば01

04H番地を指定した場合は、0100H~0117H番地の内容を表示します。(0100H~010FHの16バイト分を単位として、この16バイト分を含む24バイト分の内容を表示します。)

- [▼] を押せば、次の単位の16バイト分からのメモリ内容が表示されます。[▲] を押せば、前の単位の16バイト分からのメモリ内容が表示されます。
- 書式(2)を実行すると、以前にDコマンドで表示していた次の単位の先頭番地から表示します。初期値は0100Hになります。プリントモード時は印字します。
- 書式(3)を実行すると、プリントモード時は開始番地~終了番地のブロックのメモリ内容を印字します。印字が終了すれば、コマンド待ちの状態に戻ります。  
プリントモードでないときは、開始番地から24バイト分のメモリの内容を表示します。(この場合、終了番地の指定は意味がなくなります。)
- プリントモードの指定/解除は、Pコマンドまたは [SHIFT] + [P↔NP] で行います。
- [BREAK] を押せばコマンド (命令) 待ちの状態に戻ります。

チェックサム…**検査合計** データ項目の集まりの合計であって、そのデータが記録されるとき計算され、検査の目的でその集まりにつけられるもの。本機では、Dコマンドの実行によって表示された16バイト分のメモリの内容を合計し、その合計値の下位1バイトをチェックサムとして表示します。

プログラムをマニュアル操作で入力したときなど、このチェックサムが元のプログラムと一致することを確認することにより、表示された16バイトの中に入力誤りがないかどうかを確認できます。ただし、入力誤りが2カ所以上あった場合は、偶然チェックサムが一致してしまうことがあります。

## \* E……………イクスチェンジメモリ

**機 能** メモリの内容を書き換えます。

**書 式** (1) E 開始番地

(2) E

**説 明** ● メモリ内容の書き換えは、Sコマンドでも行えますが、Eコマンドを使用すると便利に書き換えができます。

- 書式(1)を実行すると、指定した開始番地を含む24バイト分を表示し、指定した番地でカーソルが点滅します。指定できる範囲は0000H~7FFFHまでです。
- カーソル移動キー ([←] [→] [▲] [▼]) でカーソルを移動させ、16進数で入力を行います。データを入力すると、自動的に次の番地にカーソルが移動しますので、続けて入力できます。

- 16進数のA~Fはアルファベットキーで入力しますが、右のように四則演算キーでも入力できます。

|   |     |     |     |
|---|-----|-----|-----|
| 7 | 8   | 9   | /   |
|   |     |     | (F) |
| 4 | 5   | 6   | *   |
|   |     |     | (E) |
| 1 | 2   | 3   | -   |
|   |     |     | (D) |
| 0 | .   | =   | +   |
|   | (A) | (B) | (C) |




- **[TAB]** を押すとカーソルは右側の文字表示に移動し、直接データをアスキー文字(数字および英文字など)で入力することができます。もう一度 **[TAB]** を押すと、左側の16進数でのコード入力に移ります。
- 書式(2)を実行すると、以前にEコマンドで表示していた次の単位の先頭番地から表示します。初期値は0100Hになります。
- Eコマンドを実行すると“カナ”モードは解除されます。

### \* P……………プリントスイッチ

**機 能** プリントモードの設定／解除を行います。

**書 式** P 

- 説 明**
- P  と押すたびに、プリントモードの設定と解除が交互に行われます。(プリントモードでは画面右下に“PRINT”が点灯します。)
  - **[SHIFT] + [P↔NP]** の操作と同じです。
  - プリンタが接続されていないときや、プリンタの電源が入っていないときは、Pコマンドは動きません。

### \* G……………ゴースブ

**機 能** 指定した開始番地から機械語プログラムを実行します。

**書 式** G 開始番地 

- 説 明**
- BASICのGOSUB命令と同様で、指定した開始番地から機械語プログラムを実行し、機械語のリターン命令があればコマンド待ちの状態に戻ります。
  - 必ずプログラムの最後にRET（リターン）命令を入れてください。RET命令がないと暴走します。
  - 暴走…でたらめな実行、または一定の実行を続けて止まらなくなる。内部の状態によって、どのような動きをするか分からず、多くの場合、機械語プログラムやBASICプログラム、データなど記憶内容を破壊します。
  - プログラムの実行を中止する場合は、リセットスイッチを押します。

(注) 機械語プログラムは1カ所でもまちがいがあると暴走することがあります。このため、BASICなど、他のプログラムが計算機に入っているときは、あらかじめ紙に書き写したり、印字したりしておいてください。


〈例〉Sコマンドの例で入力したプログラムを実行する場合

G 0100 

```
* G 0100
*
```

すぐにコマンド待ちの状態に戻ります。

010FH番地に01Hが入っていますので、Dコマンドで確認します。

D 0100 

```
0100 : 3E 01 18 04 >...
(1E) 3A 0F 01 3C :...^
      32 0F 01 C9 2.../
      31 00 00 01 1...
```


↑  
01Hが入っています

次に0104Hから実行します。

**[BREAK]** G 0104 

```
* G 0104
*
```

010FH番地に1が加えられています。Dコマンドで確認します。

D 0100 

```
0100 : 3E 01 18 04 >...
(1F) 3A 0F 01 3C :...<
      32 0F 01 C9 2.../
      31 00 00 02 1...
```


↑  
1が加えられています

繰り返し0104H番地から実行すれば、その都度010FH番地に1が加えられます。

### \* R……………リードSIO

**機 能** SIO（シリアル入出力装置）に送られてくるデータを読み込みます。パソコン等との機械語の通信等に使用します。

**書 式** (1) R 


(2) R 開始番地 

**説 明** SIOからインテル・ヘキサ形式で送られてくるデータを読み込みます。

- 書式(1)では、W命令実行時の番地に読み込みます。
- 書式(2)は読み込んだデータを指定した開始番地から順番に入れていきます。
- 読み込みが終了すると、データが入ったエリア（領域）を表示します。
- 読み込みを中止するときは、コマンド待ちの状態になるまで、**[BREAK]** を押し続けてください。
- 通信条件の設定は、TEXTモードのSIOで行ってください。

### \* W……………ライトSIO

**機 能** SIO（シリアル入出力装置）にデータを出力します。パソコン等との機械語の通信等に使用します。


**書 式** W 開始番地, 終了番地 

- 説 明**
- 開始番地から終了番地までのメモリ内容をSIOからインテル・ヘキサ形式で出力します。
  - 出力を中止するときは、コマンド待ちの状態になるまで、**[BREAK]** を押し続けてください。
  - 通信条件の設定は、TEXTモードのSIOで行ってください。

(注) 周辺機器接続端子(11ピン)にプリンタを動作状態で接続し、このWコマンドを実行すると本機およびプリンタが誤動作することがあります。この場合は、プリンタの電源を切り、本機の**[BREAK]** を押し続けてください。

## \*BP.....ブレークポイント

**機能** 指定したアドレス（番地）にブレークポイントを設定します。

**書式** (1) BP 番地 [—カウント数] 

(2) BP 

(3) BP 0 

**説明** ● 書式(1)を実行すると、指定した番地にブレークポイントが設定されます。

ブレークポイントは4カ所まで設定できます。4カ所設定するときは、書式(1)と同様の方法で、1カ所ずつアドレスを設定します。指定できる番地の範囲は0000H～7FFFHです。

● カウント数は0～255の範囲で指定できますが、0を指定すると、ブレークポイントの解除になります。カウント数を省略した場合は1の指定とみなします。

● 同じ番地でもカウント数が異なる場合は、別々のブレークポイントとみなします。

カウント数を0として解除した場合は、同じ番地のブレークポイントすべてを解除します。

● ブレークポイントを同時に設定できるのは4カ所までです。5カ所以上設定しようとすると、先に設定したブレークポイントから解除されます。

● ブレークポイントは命令（オペコード）の番地に設定してください。オペランドの番地に設定すると、実行が停止しないだけでなく、プログラムが正しく実行されません。

● 書式(2)では、設定されているブレークポイントのアドレスとカウント数を表示します。設定されていない場合は、次の行にプロンプト（\*）だけを表示します。

● 書式(3)では、設定されているブレークポイントすべてを解除します。特定の番地のみ解除するときは、書式(1)でカウント数を0にします。

● 1つのブレークポイントは、1回実行すると無効になります。したがって、繰り返しループの中では、指定したカウント数のときにブレークポイントとして1回だけ有効になります。ただし、Gコマンドを実行すれば、再び指定したカウント数のときに有効になります。

● 機械語モニタ以外のモードから、機械語モニタモードにしたときでも、以前に設定したブレークポイントは記憶されており、Gコマンドを実行すると有効になります。

〈例〉次の機械語プログラムが0100H～010DH番地に格納されているものとして、ブレークポイントを0105H番地と0106H番地に設定して実行してみましょう。（ブレークポイントは、命令がある番地に設定します。）

| 〈アドレス〉 | 〈機械語コード〉 | 〈ニモニック〉         | 〈プログラムの意味〉              |
|--------|----------|-----------------|-------------------------|
| 0100   | 3E       | LD A, 20H       | Aレジスタに20H（16進数）を入れる     |
| 0101   | 20       |                 |                         |
| 0102   | 21       | LD HL, 0400H    | HLレジスタに0400Hを入れる        |
| 0103   | 00       |                 |                         |
| 0104   | 04       |                 |                         |
| 0105   | 77       | LBL: LD (HL), A | HLの内容で示す番地のメモリにAの内容を入れる |
| 0106   | 3C       | INC A           | Aの内容に1を加えてAに入れる         |

|      |    |            |                                              |
|------|----|------------|----------------------------------------------|
| 0107 | 23 | INC HL     | HLの内容に1を加えてHLに入れる                            |
| 0108 | FE | CP 0A0H    | Aの内容とA0Hを比較(Aの内容からA0Hを減算)                    |
| 0109 | A0 |            |                                              |
| 010A | C2 | JP NZ, LBL | 前の計算で結果が0でない(Aの内容がA0Hでない)とき、ラベルLBLのある番地へジャンプ |
| 010B | 05 |            |                                              |
| 010C | 01 |            |                                              |
| 010D | C9 | RET        | リターン                                         |

機械語モニタモードにして、0105H番地にブレークポイントを設定します。

 MON 


BP 0105 

0106H番地にブレークポイントを設定します。

BP 0106-3 

ブレークポイントが2カ所に設定されました。

次にGコマンドで0100H番地から実行します。

G 0100 

ブレークポイントを設定したアドレスで実行を停止し、レジスタの内容を表示します。

実行を再開させます。



2カ所目のブレークポイント設定アドレスで停止します。

```
MACHINE LANGUAGE MONITOR
*BP 0105
BP=0105 ( 1)
*
```

```
MACHINE LANGUAGE MONITOR
*BP 0105
BP=0105 ( 1)
*BP 0106-3
BP=0105 ( 1), 0106 ( 3)
*
```

```
PC=0105 AF=20 44
SP=7FF6 BC=00 00
IX=7C06 DE=00 00
IY=7C03 HL=04 00
```

PC（プログラムカウンタ）の値が、停止したアドレスを示します。  
Aレジスタに20Hが入っています。


```
PC=0106 AF=22 87
SP=7FF6 BC=00 00
IX=7C06 DE=00 00
IY=7C03 HL=04 02
```

0106H番地で停止。  
Aレジスタには2回カウントアップした結果22Hが入っています。

このように、機械語プログラムを変更することなく、ブレークポイントの設定ができます。

(注) ・ブレークポイントを設定すると、設定アドレスの内容を一時F7Hに置き換えてプログラムを実行します。

ブレークポイントを設定した状態で、プログラム実行中にリセットスイッチ（RESET）を押すと、そのとき実行していないブレークポイント設定アドレスのメモリ内容が、F7Hに置き換えられたままになります。このような場合は、ブレークポイントを設定したアドレスの内容を確認して、F7Hになっているときは元の値に書き直す必要があります。

・ブレークポイントが不要になったときは、書式(3)（BP 0 ）を実行して解除してください。  
設定されたブレークポイントは、モードを切り替えたり、機械語プログラムを書き換えても保

持されています。そして、Gコマンドで機械語プログラムを実行するたびに、設定アドレスで停止します。

- メモリ内容がF7Hのアドレスには、ブレークポイントを設定できません。

### 3. 機械語モニタモードでのエラー表示

機械語モニタモードでは以下のエラー表示があらわれます。

| エラーメッセージ         | 内 容                                   |
|------------------|---------------------------------------|
| SYNTAX ERROR     | 文法的に実行できない。                           |
| MEMORY ERROR     | 機械語エリアとして使用できる範囲を外れて、機械語エリアを確保しようとした。 |
| I/O DEVICE ERROR | SIOに対するリードインエラー、チェックサムエラーなど。          |
| OTHER ERROR      | その他のエラー。                              |

## アセンブラ機能

### 1. ソースプログラムと機械語プログラム

機械語は、コンピュータが直接、解読・実行できる唯一の言語です。

機械語でプログラムを作れば、BASICなどに比べて処理スピードが速くなり、コンピュータの機能を最大限に活用できます。

しかし、この機械語はすべて数字で表され、人間にとっては大変覚えにくく、扱いにくい言語です。このため、機械語のプログラムを作成するときは、先に各命令を覚えやすい符号に置き換えて（符号化して）プログラムを作成した後、符号を機械語コードに変換して機械語プログラムを完成させる方法が用いられます。

符号化した命令（ニモニック）で書かれたプログラムをソースプログラムと呼びます。

これに対して、変換して得られた機械語プログラムをオブジェクトプログラムと呼びます。（274ページの参考を参照）

| 〈例〉 | 〈ニモニックで書かれたプログラム〉 | 〈機械語プログラム〉<br>(オブジェクトプログラム) | 〈プログラムの意味〉                 |
|-----|-------------------|-----------------------------|----------------------------|
|     | LD A, 2           | 3 E 0 2                     | Aレジスタに2を入れなさい              |
|     | LD B, 3           | 0 6 0 3                     | Bレジスタに3を入れなさい              |
|     | ADD A, B          | 8 0                         | AとBの内容を加えて、Aレジスタに入れなさい     |
|     | LD (010FH), A     | 3 2 0 F 0 1                 | 010FH番地のメモリにAレジスタの内容を入れなさい |
|     | RET               | C 9                         | 戻りなさい（サブルーチン・リターン）         |

この例のように、ニモニックで書かれたプログラムは、機械語プログラムに比べるとわかりやすいので、プログラムが作りやすくなります。

ニモニックで書かれたプログラムをコンピュータで実行させるためには、機械語プログラムに変換しなければなりません。この変換作業をアセンブルと呼びます。

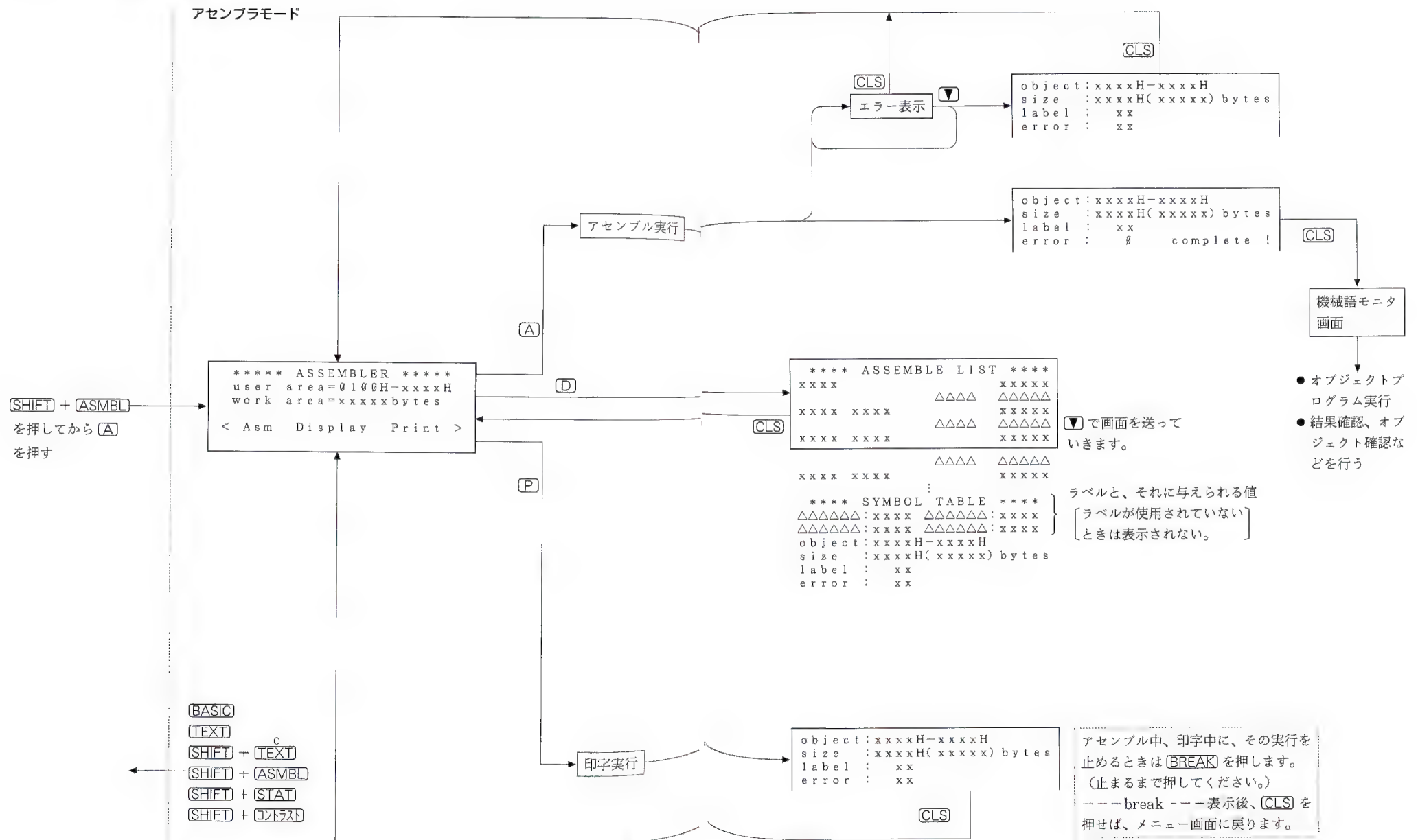
人間が手操作でアセンブルすることをハンドアセンブルと呼びます。上例のような短いプログラムなら、ハンドアセンブルしてもすぐに終わりますが、長いプログラムをハンドアセンブルするのは大変な作業で、まちがえることもあります。そこで登場するのが、アセンブラです。

アセンブラは、ニモニックで書かれたプログラムを機械語プログラムに翻訳（変換）するプログラムです。これを使えば大変速く、アセンブルするときのまちがいがなくなります。

本機は、Z80機械語用のアセンブラを持っています。以降に、このアセンブラの使いかたを説明します。

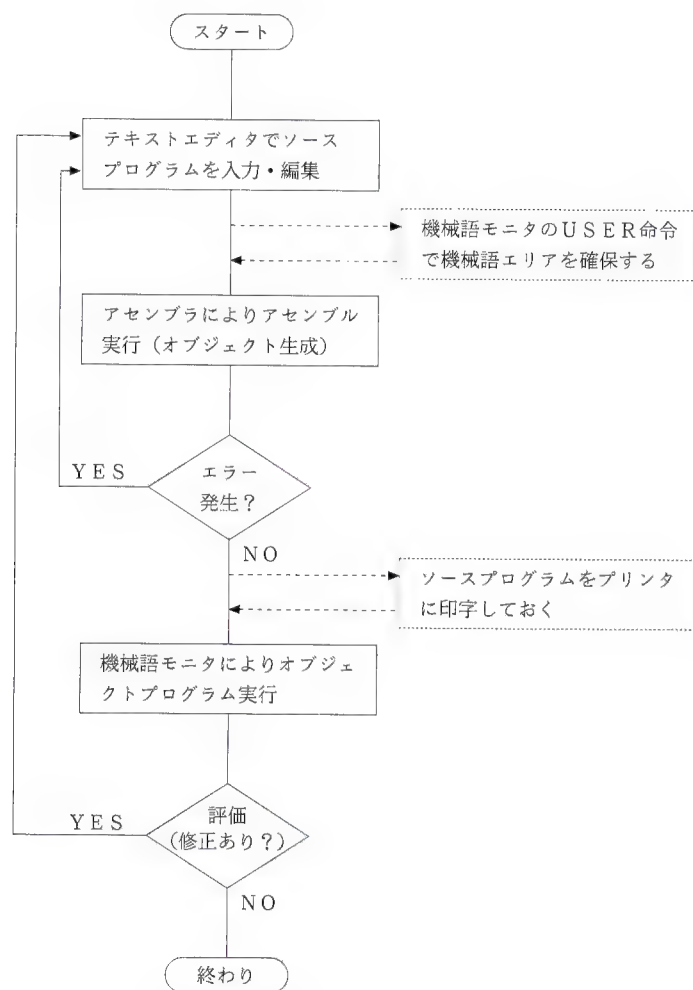
## 2. アセンブラの機能

### アセンブラ機能一覧





## アセンブラの手順



## 3. アセンブルしてみましょう

アセンブラやソースプログラムを説明する前に、とりあえずプログラム例を入れて、アセンブルしてみましょう。そして、できたオブジェクトプログラム（機械語プログラム）を実行してみましょう。

機械語プログラムを実行する前に知っておいていただきたいことがあります。

機械語プログラムを実行したとき、プログラムや操作に誤りがあると、すべてのプログラムやデータが消えることがあります（273ページ参照）。したがって、消えては困るプログラムやデータは、紙に書き写したり、印字しておいてください。

〈プログラム例〉

0400H～047FH番地のメモリに20Hから9FHまでの数値を入れるプログラム（Hは16進数を示す記号（文字）です）。

〈ソースプログラム〉

|     |        |              |                                               |
|-----|--------|--------------|-----------------------------------------------|
| 10  | ORG    | 0100H        | （オブジェクトを0100H番地以降に入れない）                       |
| 20  | START: | LD A, 20H    | Aレジスタに20Hを入れる                                 |
| 30  |        | LD HL, 0400H | HLレジスタに0400Hを入れる                              |
| 40  | LBL:   | LD (HL), A   | HLの内容で示す番地のメモリにAの内容を入れる                       |
| 50  |        | INC A        | Aの内容に1を加えてAに入れる                               |
| 60  |        | INC HL       | HLの内容に1を加えてHLに入れる                             |
| 70  |        | CP 0A0H      | Aの内容と0A0Hを比較（Aの内容から0A0Hを減算）                   |
| 80  |        | JP NZ, LBL   | 前の計算で結果が0でない（Aの内容が0A0Hでない）とき、ラベルLBLのある番地へジャンプ |
| 90  |        | RET          | リターン（サブルーチンリターン）                              |
| 100 |        | END          | （ソースプログラム終わり）                                 |

〈オブジェクト〉

```

3E 20
21 00 04
77
3C
23
FE A0
C2 05 01 ←ラベルは番地に変換されます。（0105H番地）
C9
  
```

ソースプログラムの10行と100行の命令は、疑似命令と呼ばれ、アセンブラを制御する命令です。それ自身は機械語（オブジェクト）に変換されません。（300ページ参照）

## (1) ソースプログラムの入力

それでは、ソースプログラムを入力してみましょう。ソースプログラムはTEXTプログラムとして入力します。(169ページ参照)

**[TEXT]** を押してテキストモードの機能選択画面  
(メインメニュー画面) にします。

```
*** TEXT EDITOR ***
Edit Del Print
Sio File Basic Rfile
```

**[E]** を押して、エディット機能を選びます。

```
TEXT EDITOR
<
```

すでにTEXTプログラムが入っているときは、  
機能選択画面で **[D]** を押した後、**[Y]** を押して消  
去してください。

ソースプログラムを入力します。

| [キー操作]                                                    | [表 示]                  |
|-----------------------------------------------------------|------------------------|
| 1 0 <b>[TAB]</b> ORG <b>[TAB]</b> 0 1 0 0 H <b>[↵]</b>    | 1 0 ORG 0 1 0 0 H      |
| 2 0 START: LD <b>[TAB]</b> A, 2 0 H <b>[↵]</b>            | 2 0 START: LD A, 2 0 H |
| 3 0 <b>[TAB]</b> LD <b>[TAB]</b> HL, 0 4 0 0 H <b>[↵]</b> | 3 0 LD HL, 0 4 0 0 H   |
| 4 0 LBL: <b>[TAB]</b> LD <b>[TAB]</b> (HL), A <b>[↵]</b>  | 4 0 LBL: LD (HL), A    |
| 5 0 <b>[TAB]</b> INC <b>[TAB]</b> A <b>[↵]</b>            | 5 0 INC A              |
| 6 0 <b>[TAB]</b> INC <b>[TAB]</b> HL <b>[↵]</b>           | 6 0 INC HL             |
| 7 0 <b>[TAB]</b> CP <b>[TAB]</b> 0 A 0 H <b>[↵]</b>       | 7 0 CP 0 A 0 H         |
| 8 0 <b>[TAB]</b> JP <b>[TAB]</b> NZ, LBL <b>[↵]</b>       | 8 0 JP NZ, LBL         |
| 9 0 <b>[TAB]</b> RET <b>[↵]</b>                           | 9 0 RET                |
| 1 0 0 <b>[TAB]</b> END <b>[↵]</b>                         | 1 0 0 END              |

入力が終わったら、まちがいがいいことを確認してください。

アセンブルの前にオブジェクトを格納するための機械語エリアを確保しておきます。確保しておかないとアセンブルできません。

## (2) 機械語エリアの確保

機械語エリアは機械語モニタのUSER命令で確保します。(275ページを参照)

機械語モニタモードにします。

**[BASIC] MON [↵]**

```
MACHINE LANGUAGE MONITOR
*
```

USER命令で機械語エリアを確保します。

ここでは、0 4 FFHまで(0 1 0 0 H~0 4 FFH)を確保しておきます。

USER 0 4 FF **[↵]**

```
MACHINE LANGUAGE MONITOR
*USER 0 4 FF
FREE: 0 1 0 0 0 4 FF
*
```

確保した機械語エリア(ユーザーエリア)が表示されます。

## (3) アセンブルの実行

ソースプログラムを機械語プログラムに変換します。

**[SHIFT] + [ASMBL]** と押してから **[A]**

アセンブラモードになります。

(画面のワークエリアの値は変わることがあります。295ページを参照してください。)

```
***** ASSEMBLER *****
user area=0 1 0 0 H-0 4 FFH
work area= 4 6 4 5 bytes

< Asm Display Print >
```

**[A]**

アセンブルが実行されます。

```
***** ASSEMBLER *****

--- assembling ---
```

アセンブルが正常に終われば右のように表示されます。(画面の内容については296ページ参照)

```
object: 0 1 0 0 H-0 1 0 DH
size : 0 0 0 EH( 14) bytes
label : 2
error : 0 complete !
```

アセンブル中にエラーが発生したときは、右のようにエラーの内容と、エラーが発生した行が表示されます。(296ページ参照)  
このときはテキストエディタへ戻ってソースプログラムを修正します。

```
***** ASSEMBLER *****

*FORMAT ERROR (1)
0 1 0 5 ***** 4 0
LBL: LD HL), A
```

## (4) 生成したオブジェクトの確認

生成したオブジェクトを機械語モニタで確認してみましょう。オブジェクトは0 1 0 0 H~0 1 0 DH番地に入っています。

**[CLS]** を押します。

(または **[BASIC] MON [↵]**)

機械語モニタモードになります。

D命令(ダンプメモリ)で呼び出します。

D 0 1 0 0 **[↵]**

オブジェクトが表示されます。

```
MACHINE LANGUAGE MONITOR
*
```

```
0 1 0 0 : 3E 20 21 00 > !.
(88) 04 77 3C 23 . w<#
FE A0 C2 05 . ツ.
01 C9 00 00 . ノ..
0 1 1 0 : 20 00 19 00 . .
00 00 00 00 . . . .
```

(0 1 0 EH番地以降は前に記憶していた内容が残っています。)

## (5) オブジェクト(機械語)プログラムの実行

生成したオブジェクトプログラムを実行してみましょう。機械語モニタのG命令(ゴーサブ)で実行します。

機械語モニタの命令待ち状態に戻します。

**[BREAK]**

実行します。

G 0 1 0 0 **[↵]**

実行が終わると命令待ち状態に戻ります。

結果を確認しましょう。

D 0 4 0 0 **[↵]**

**[▼]**

0 4 0 0 H ~ 0 4 7 F H 番地に 2 0 H ~ 9 F H ま  
での16進数値が入っています。

|             |   |     |     |     |     |     |   |   |   |
|-------------|---|-----|-----|-----|-----|-----|---|---|---|
| *           |   |     |     |     |     |     |   |   |   |
| * G 0 1 0 0 |   |     |     |     |     |     |   |   |   |
| *           |   |     |     |     |     |     |   |   |   |
| 0 4 0 0     | : | 2 0 | 2 1 | 2 2 | 2 3 | !   | " | # |   |
| (7 8)       |   | 2 4 | 2 5 | 2 6 | 2 7 | \$  | % | & | ' |
|             |   | 2 8 | 2 9 | 2 A | 2 B | ( ) | * | + |   |
|             |   | 2 C | 2 D | 2 E | 2 F | ,   | - | / |   |
| 0 4 1 0     | : | 3 0 | 3 1 | 3 2 | 3 3 | 0   | 1 | 2 | 3 |
|             |   | 3 4 | 3 5 | 3 6 | 3 7 | 4   | 5 | 6 | 7 |
|             |   |     |     |     |     |     |   |   |   |
| 0 4 1 0     | : | 3 0 | 3 1 | 3 2 | 3 3 | 0   | 1 | 2 | 3 |
| (7 8)       |   | 3 4 | 3 5 | 3 6 | 3 7 | 4   | 5 | 6 | 7 |
|             |   | 3 8 | 3 9 | 3 A | 3 B | 8   | 9 | : | : |
|             |   | 3 C | 3 D | 3 E | 3 F | <   | = | > | ? |
| 0 4 2 0     | : | 4 0 | 4 1 | 4 2 | 4 3 | @   | A | B | C |
|             |   | 4 4 | 4 5 | 4 6 | 4 7 | D   | E | F | G |

## 4. ソースプログラムの作成・編集

本機のアセンブラでは、CASLのアセンブルと同様、TEXTエリアに入れたソースプログラムをオブジェクトコードに変換（アセンブル）します。

変換したオブジェクトコードは、指定した番地のメモリ以降へ順番に入れています。

ここでは、ソースプログラムを作成するための仕様（入力形式など）について説明します。

### 4.1 ソースプログラムの構成

ソースプログラムの1行は通常、1ステートメント（1つの文）になります。

この行が何行か集まって、1つのプログラムが構成されます。

1つのソースプログラムはORG命令から始まって、END命令で終わります（ただし、この命令は省略可能です）。

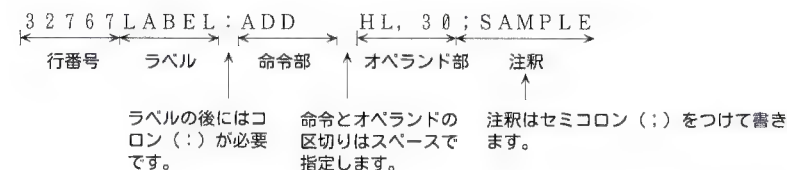
```
<例> 1 0      ORG      0 1 0 0 H
      }
      1 0 0      END
```

- ORG命令は、生成したオブジェクトを入れる番地を指定します。つまり、生成したオブジェクトは、ORG命令で指定した番地から順番に入れられます。
- END命令は、ソースプログラムの終わりを指定します。つまり、END命令があると、そこでアセンブルを終了します。

これらの命令は、アセンブラを制御する命令ですので、オブジェクトには変換されません。（300ページの疑似命令参照）

### ◆行（ステートメント）の構成

1行は、行番号、ラベル、命令、オペランド、注釈、疑似命令などにより構成されます。



- 1行の長さは、注釈を含めて最大254文字までです。
- アルファベットの小文字は、オペランドで文字列として書いた場合と、注釈に書いた場合以外は、大文字と同じものとみなします。

#### ①行番号（ラインナンバー）

- 行番号は1～65279までの数値が使用できます。
- 1～65279の範囲外の値を指定すると“LINE NO. ERROR”と表示されます。

#### ②ラベル

- ラベルは行番号に続けて入力します。（ラベルと行番号との間にスペースを入れるとエラーになります。）
- ラベルは1～6文字まで使用できます。7文字以上入れるとエラーになります。
- ラベルには次の文字が使用できます。

英文字（A～Z）、数字（0～9）、記号（[、]、@、?、\_）。

ただし、数字はラベルの先頭には使えません（行番号と区別ができなくなります）。

- 次のレジスタおよび条件コードと同じ文字（文字列）は単独で使えません。

（シングルレジスタ） A、B、C、D、E、H、L、I、R

（ペアレジスタ） AF、BC、DE、HL、IX、IY、SP

（条件コード） NZ、Z、NC、C、PO、PE、P、M

- ラベルの後ろにはコロン（:）が必要です。コロンがないとエラーになります。

ただし、疑似命令のEQU命令で、ラベルに値を定義するときはコロンをつけません。

- ラベルを書かないときは、行番号と次の命令の間に1つ以上のスペース（空白）を入れてください。

**[TAB]**を用いることもできます。

#### ③命令部（オペコード）

- Z80の命令をニモニックコードで書き表します。また、300ページ以降に記載している疑似命令も使います。なお、命令をオペコード（オペレーションコード）と呼びます。

- 命令部と、次のオペランドとの間は1つ以上のスペース（空白）で区切ります。

**[TAB]**を用いることもできます。

#### ④オペランド部

命令（オペコード）に対して、その実行対象となるレジスタ、アドレス（番地）、定数などをオペランドといいます。

- オペランドが2つ以上あるときは、コンマ（,）で区切ります。

- 1つのオペランドは、32文字まで書くことができます。
- 定数には次のものが使用できます。

#### 【数値定数】

2進数、10進数、16進数が使用できます。

2進数…… 0と1の数字で表し、最後にBをつけます。

〈例〉 1 0 1 1 1 1 0 0 B、1 0 0 0 0 0 B

10進数…… 0～9の数字で表します。

〈例〉 1 8 8、3 2

16進数…… 0～9、A、B、C、D、E、Fの16進用数字で表し、最後にHをつけます。数値がA、B、C、D、E、Fで始まる場合は、先頭に0をつけて他の命令と区別できるようにします。

〈例〉 0 B C H、2 0 H

#### 【文字定数（文字列）】

文字列は'（シングルクォーテーション）で囲んで表します。文字のアスキーコードが定数になります。

| 〈例〉    | （文字列）       | （指定） | （定数）              |
|--------|-------------|------|-------------------|
| A      | ' A '       |      | 4 1 H             |
| A B    | ' A B '     |      | 4 1 H、4 2 H       |
| B ' C  | ' B ' ' C ' |      | 4 2 H、2 7 H、4 8 H |
| ' D    | ' ' ' D '   |      | 2 7 H、4 4 H       |
| E '    | ' E ' ' '   |      | 4 5 H、2 7 H       |
| '      | ' ' ' ' '   |      | 2 7 H             |
| (NULL) | ' '         |      | 0 0 H             |

#### 【ラベル定数】

EQU命令により、ラベルに定数を定義しておけば、そのラベルを定数として使用できます。

- 数式（四則計算）が使用できます。

次の符号と演算子が使用できます。ただし、計算の優先順位の判別は行いません。

（符 号） +、-

（演算子） \*、/、+、-

- 計算は16ビットで行われます。オーバーフロー（桁あふれ）が起こっても、オーバーフロー分は無視します（エラーにはなりません）。計算結果の8ビットまたは16ビットを使用してオブジェクトを生成します。
- 数式が書けるところでは、バイト、ワードのチェックは行いません。

〈例〉 LD A, 4 1 4 2 H → LD A, 4 2 H とみなされます。  
DB 1 2 3 4 H → DB 3 4 H とみなされます。

#### ⑤注釈（コメント）

各行はセミコロン（;）をつけて注釈を書くことができます。セミコロン以降は、行の終わり（行末）まで注釈とみなされ、機械語（オブジェクト）には変換されません。

## 4.2 ソースプログラムの消去

TEXTモードの機能選択画面で、**[D]**を押せばデリート（Del）機能が選ばれ、次のようにテキスト内容を消去（削除）してよいかが聞いてきます。（テキスト内容がない場合は、画面は変わりません。）

TEXT DELETE OK? (Y)

**[Y]**を押せば、テキスト内容がすべて消去され、TEXTモードの機能選択画面に戻ります。

くわしくは174ページを参照してください。

## 4.3 ソースプログラムの入力

TEXTモードの機能選択画面で **[E]**を押せばエ

ディット機能が選ばれます。

TEXT EDITOR  
<

次にソースプログラムの入力手順を説明します。

①行番号を入力します。

②ラベルがないときは、**[TAB]**を押します。カーソルが命令部に移ります。

**[TAB]**の代わりに **[SPACE]** でスペースを入力してもかまいません。

スペースは1つ以上入れてください。

ラベルを入力するときは、行番号に続けて入力し、コロン（:）を入れます。この後はスペースを入れても、入れなくてもかまいません。

③命令を入力します。

命令の後にオペランドがある場合は、**[TAB]** または **[SPACE]** でスペースを1つ以上入れます。

④オペランドを入力します。

オペランドが2つ以上ある場合は、コンマ（,）で区切って入力します。

⑤注釈をつける場合は、セミコロン（;）を入力して、その後に入力します。

⑥1行の入力を完了したら **[↵]**を押して、プログラムをメモリに格納します。**[↵]**を押せばカーソルが消えます。

次の行を入力するときは、①から繰り返します。

289ページのソースプログラムの入力例を参照してください。

# 5. アセンブル

TEXT（テキスト）モードで入力したソースプログラムをアセンブル（オブジェクトに変換）します。288ページのプログラム例が入力されているものとして説明します。プログラムを入力しておいてください。

## 5.1 メニュー画面

アセンブルする場合、まずアセンブラモードにします。



▼を押せば次へ進みます。

```
***** ASSEMBLER *****
*UNDEFINED SYMBOL
0100 ***** 80
          JP  NZ, KB
          L
エラー内容 (定義されていないラベルを使っている)
```

▼で次へ進みます。

アセンブルの終了画面になります。

このときは、“complete!”は表示されません。

```
object:0100H-010CH
size  :000DH( 13)bytes
label : 2
error : 2
```

この後、**[CLS]**を押すと、アセンブラのメニュー画面に戻ります。

- この例では、50行の命令はないものとみなされ、80行のラベルは0000H番地の指定とみなされて、アセンブルされます。
- エラーが検出された場合、生成されたオブジェクトプログラムは、正しいプログラムにはなりません。これを実行すると暴走したり、記憶内容を壊したりする場合があります。ソースプログラムを修正して、再度アセンブルしてから、エラーのないプログラムを実行してください。

### 5.3 アセンブルリストを表示させる方法

ソースプログラムをアセンブルした場合に、各行で生成されるオブジェクトおよび格納されるアドレスなどを確認することができます。

アセンブラのメニュー画面で、**[D]**を押せばアセンブルリストの表示が開始されます。

以降▼で表示を送っていきます。

288ページのプログラム例が入っているものとして、実行してみましょう。

メニュー画面で**[D]**を押します。

```
***** ASSEMBLE LIST *****
0100          ORG  0100H
0100 3E20      START:LD  A, 20H
0102 210004
          ソースプログラム
```

オブジェクト欄が空白の場合は、オブジェクトがありません。8桁を超えるときは8桁ごとに行を変えて表示します。

以降▼で表示を送って確認していきます。

```
LD  HL, 04
0105 77      LBL: LD  (HL),
          A
0106 3C      INC  A
0107 23      INC  HL
0108 FEA0     CP   0A0H
010A C20501   JP   NZ, LBL
010D C9      RET
010E      END

***** SYMBOL TABLE *****
START :0100 LBL :0105
object:0100H-010DH
size  :000EH( 14)bytes
label : 2
error : 0 complete !
```

シンボルテーブル※  
オブジェクト格納番地  
オブジェクトの大きさ  
ラベル数  
エラー件数

※シンボルテーブルは、ラベルに与えられる値を16進数で示します。

- リスト表示中に**[CLS]**を押せば、メニュー画面に戻ります。
- リスト表示では、生成したオブジェクトを機械語エリアに格納しません。格納するには、アセンブルを行ってください。

### 5.4 アセンブルリストを印字させる方法

C E-126 P (プリンタ)をお持ちの場合は、本機に、プリンタ C E-126 Pを接続してプリンタの電源を入れ、アセンブラのメニュー画面で**[P]**を押すと、アセンブルリストが印字されます。

- プリンタの電源が入っていない場合や接続されていない場合などには、**[P]**を押したとき“\* PRINTER ERROR”と表示されます。このときは**[CLS]**でエラーを解除し、プリンタを確認してください。
- 画面右下の“PRINT”が点灯していなくても、リストは印字できます。リストの印字が終了するとアセンブル終了画面になります。**[CLS]**を押せばメニュー画面に戻ります。

印字の様式はリスト表示の様式と同じです。

〈印字例〉

```

**** ASSEMBLE LIST ****
0100          10
          ORG  0100H
0100 3E20          20
          START:LD  A,20H
0102 210004          30
          LD  HL,04
          00H
0105 77          40
          LBL: LD  (HL),
          A
0106 3C          50
          INC  A
0107 23          60
          INC  HL
0108 FEA0          70
          CP   0A0H
010A C20501          80
          JP   NZ,LB
          L
010D C9          90
          RET
010E          100
          END

**** SYMBOL TABLE ****
START :0100 LBL  :0105
object:0100H-010DH
size  :000EH( 14)bytes
label : 2
error : 0  complete !

```

- リストの印字を途中で中止させる場合は、印字が止まるまで **BREAK** を押してください。

印字を中止すると “--- break ---” と表示されます。**CLS** を押せば、メニュー画面に戻ります。

アセンブラのメニュー画面で **□** を押すと、ミニ I/O のパラレルポートからアSEMBルリストが送出されます。詳しくは、先生の指導に従ってください。(366ページを参照)

## 6. アセンブラの疑似命令の説明

疑似命令は、アセンブラを制御する命令で、それ自体はオブジェクトに変換されません。

本機のアセンブラには、次の疑似命令があります。

- 格納開始番地の指定  
ORG 命令
- データの定義  
DEFB/DB/DEFM/DM 命令  
DEFS/DS 命令  
DEFW/DW 命令
- ラベルの値の定義  
EQU 命令
- アセンブル終了の指定  
END 命令

次に、これらの疑似命令の機能を説明します。

〈書式で使用している用語、記号の意味〉

式          数値、数式、ラベル、' 文字列' を書くことができます。

数式          数値、ラベル、およびこれらを用いた四則計算式を書くことができます。

{ }          { } 内に併記されている内容のいずれか1つを選ぶことを示します。

[ ]          [ ] で囲まれた部分の省略ができることを示します。

[ ] … [ ] で囲まれた部分を省略、または繰り返しができることを示します。

### ORG……オリジン

**機 能** オブジェクトの格納開始番地（アドレス）および、実行番地を指定します。

**書 式** ORG [式<sub>1</sub>, ] 式<sub>2</sub>

**説 明** ● 式<sub>2</sub>により、アSEMBルをして生成されたオブジェクトの格納開始番地を指定します。

式<sub>2</sub>で指定した番地以降に、オブジェクトが順番に格納されます。

- 式<sub>1</sub>により、最終的にオブジェクトプログラムを格納し、実行する場所の開始番地を指定します。(次ページの **ご注意** を参照)

- 式<sub>1</sub>を省略した場合は、式<sub>2</sub>と同じ番地を指定したものとみなされます。

- ソースプログラムにORG命令を書かなかった場合は、ORG 1000H の設定があるものとみなして、オブジェクトの生成、格納が行われます。

- 式<sub>1</sub>は0HからFFFFHの範囲内で指定できます。ただし、オブジェクトプログラムがこの範囲を超えないように指定してください。(この範囲でも、メモリの制約によりエラーになる場合があります。)

- 式<sub>2</sub>は9 0 Hから、機械語モニタのUSER命令で確保した機械語エリアの範囲内で指定できます。ただし、オブジェクトプログラムがこの範囲を超えないように指定してください。

### ご注意

先の説明のように、アセンブルして生成したオブジェクトは式<sub>2</sub>で指定した番地から格納されます。そのまま（この場所で）オブジェクトプログラムを実行する場合、式<sub>1</sub>は式<sub>2</sub>と同じ番地にします。（このとき、式<sub>1</sub>の指定は省略できます。）

図のように、生成したオブジェクトプログラムをパソコン等に記録し、それを別の場所を読み込んで実行する場合は、読み込む場所の番地を式<sub>1</sub>で指定します。

式<sub>1</sub>と式<sub>2</sub>で別々のアドレスを指定した場合は、アセンブル実行後、必ず上図のようにオブジェクトプログラムを式<sub>1</sub>で指定した番地で始まるメモリに移し変えて、実行してください。

そのまま実行すると、プログラムによっては暴走する恐れがあります。

- 式<sub>1</sub>と式<sub>2</sub>に別々の番地が指定されると、アセンブラは、式<sub>1</sub>で指定された番地からオブジェクトを格納するものとみなしてアセンブルします（実際は、式<sub>2</sub>で指定した番地以降に格納します。）つまり、ソースプログラムのラベル等に与えられる番地は、式<sub>1</sub>で指定された番地を基準にした値になります。このため、生成されたオブジェクトが格納された番地と、プログラムに与えられた番地が異なります。したがって、このオブジェクトプログラムを式<sub>1</sub>で指定した番地に移し変えないで実行すると、暴走する可能性があります。

例 ORG 0400H

0400H番地からオブジェクトを格納します。

ORG 0C00H, 0100H

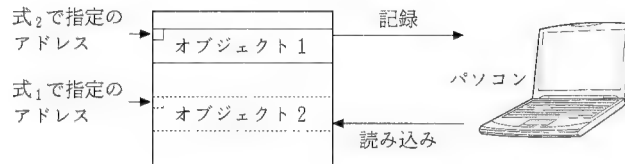
0C00H番地を開始番地としてアセンブルし、生成したオブジェクトを0100H番地以降に格納します。

### DEFB/DB/DEFM/DM……デファイン バイト/デファイン メッセージ

機能 オペランドに記述した数値の下位1バイト、または文字列をオブジェクトに生成します。

書式

|        |                                                                                                                                                                                                     |   |      |   |   |    |   |   |      |   |   |    |   |           |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|---|---|----|---|---|------|---|---|----|---|-----------|
| [ラベル:] | <table border="0"> <tr><td>{</td><td>DEFB</td><td>}</td></tr> <tr><td>{</td><td>DB</td><td>}</td></tr> <tr><td>{</td><td>DEFM</td><td>}</td></tr> <tr><td>{</td><td>DM</td><td>}</td></tr> </table> | { | DEFB | } | { | DB | } | { | DEFM | } | { | DM | } | 式 [, 式] … |
| {      | DEFB                                                                                                                                                                                                | } |      |   |   |    |   |   |      |   |   |    |   |           |
| {      | DB                                                                                                                                                                                                  | } |      |   |   |    |   |   |      |   |   |    |   |           |
| {      | DEFM                                                                                                                                                                                                | } |      |   |   |    |   |   |      |   |   |    |   |           |
| {      | DM                                                                                                                                                                                                  | } |      |   |   |    |   |   |      |   |   |    |   |           |



説明 ● オペランドに記述された数値の下位1バイトがオブジェクトに生成されます。

〈例〉 DEFB 1234H 34Hがオブジェクトになります。  
DB 1234 D2Hがオブジェクトになります。(10進数1234は16進数では4D2です。)

- 文字列は、' (シングルクォーテーション) で囲んで記述します。32文字まで記述できます。文字列は、その1文字1文字のアスキーコードがオブジェクトとして生成されます。

〈例〉 DEFM 'DATA' 44H、41H、54H、41Hがオブジェクトになります。

- オペランドが複数ある場合は、, (コンマ) で区切って記述します。

〈例〉 DB 32\*4+5, 'X2' 85H、58H、32Hがオブジェクトになります。

### 例

| 〈ソースプログラム〉 |                     | 〈オブジェクトプログラム〉 |  | アドレス | 内容 |
|------------|---------------------|---------------|--|------|----|
| 10         | ORG 0100H           |               |  | 0100 |    |
| 20         | LD HL, DATA         | 21 0C 01      |  | 0101 |    |
| 30         | LD DE, 300H         | 11 00 03      |  | 010C | 41 |
| 40         | LD BC, 5            | 01 05 00      |  | 010D | 42 |
| 50         | LDIR                | ED B0         |  | 010E | 43 |
| 60         | RET                 | C9            |  | 010F | 44 |
| 70         | DATA: DB 'ABCDEFGH' | 41 42 43 44   |  | 0110 | 45 |
|            |                     | 45 46 47 48   |  | 0111 | 46 |
|            |                     |               |  | 0112 | 47 |
|            |                     |               |  | 0113 | 48 |
| 80         | END                 |               |  | 0114 |    |
|            |                     |               |  | 0300 | 41 |
|            |                     |               |  | 0301 | 42 |
|            |                     |               |  | 0302 | 43 |
|            |                     |               |  | 0303 | 44 |
|            |                     |               |  | 0304 | 45 |
|            |                     |               |  | 0305 |    |

70行の文字列は、1字1字がオブジェクトに変換されます。

このプログラムは、ラベルDATAで示す番地以降にあるデータを5バイト分だけ、300H番地以降へコピーします。

つまり、41H、42H、43H、44H、45Hを300H～304H番地にコピーします。



### LDIR命令について

LDIRだけで次の一連の実行がなされます。くわしくはZ80に関する市販の書籍を参照してください。

- (DE) ← (HL) HLで指定されたアドレスの内容をDEで指定されたアドレスに格納します。アドレスは間接指定で、最初は010Cのアドレスの内容を300のアドレスに格納します。
- DE ← DE + 1 DE、HLレジスタの値をカウントアップします。
- HL ← HL + 1
- BC ← BC - 1 BCレジスタの値をカウントダウンします。
- Repeat until BCレジスタの値が0になるまで繰り返します。
- BC = 0

| エラーの種類・エラーメッセージ | 内 容 (原因)            |
|-----------------|---------------------|
| フォーマット エラー      | FORMAT ERROR (1)    |
|                 | FORMAT ERROR (2)    |
|                 | FORMAT ERROR (3)    |
|                 | FORMAT ERROR (4)    |
|                 | FORMAT ERROR (5)    |
|                 | FORMAT ERROR (6)    |
|                 | FORMAT ERROR (7)    |
| アンデファインドシンボル    | UNDEFINED SYMBOL    |
| マルチ デファインシンボル   | MULTI DEFINE SYMBOL |
| ファイル ノット エグジスト  | FILE NOT EXIST      |
| ユーザーエリアオーバー     | USER AREA OVER      |
| ワークエリアオーバー      | WORK AREA OVER      |
| プリンタ エラー        | PRINTER ERROR       |

# 第9章

## P I C

### 1. P I C

PIC (Peripheral Interface Controller) とは、コンピュータの周辺に接続される周辺機器の接続部分を制御するために米マイクロチップテクノロジー社が開発した“マイコン”と呼ばれる領域のICです。PICについては、入門書等の関連書籍が市販されていますので、それらの書籍を参照してください。この章では、P I C用プログラムの作成方法について説明します。なお、本機でプログラムの作成が可能なP I C (14ビット・コアFlashメモリのP I C) は次のとおりです。

2001年7月9日現在

| 品 名       | プログラムメモリ | ピン数  |
|-----------|----------|------|
| PIC16F627 | 1 Kワード   | 18ピン |
| PIC16F84  | 1 Kワード   | 18ピン |
| PIC16F84A | 1 Kワード   | 18ピン |

P I C製品名や開発ツールなどの固有名詞は米国マイクロチップ・テクノロジー社の商品名などです。

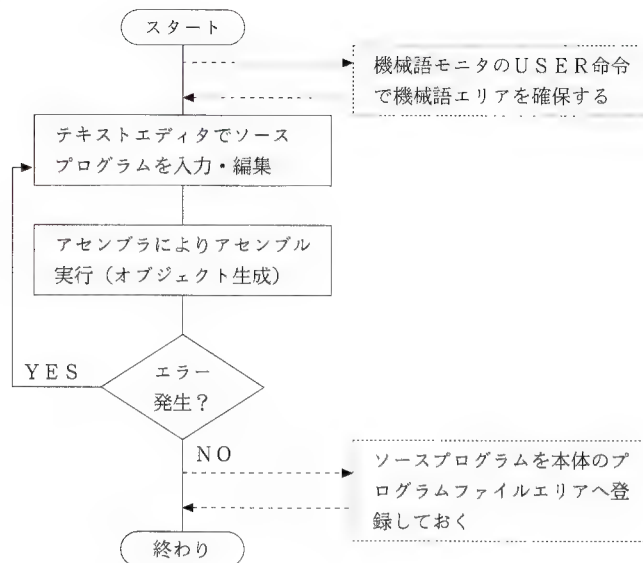
### 2. P I Cプログラムの作成手順

P I Cプログラムの作成手順とその準備について説明します。なお、P I Cプログラムの詳しい使いかたについては、先生の指導に従ってください。



## 2. 1 プログラム作成手順

P I C のプログラムの作成は次の手順で行います。



### 【作成手順の解説】

- ①モニタ機能を使用して機軸語エリアを確保します。
- ②TEXTエディタ機能を使用してソースプログラムを入力します。
- ③PICモードのアセンブラ機能によりオブジェクトプログラムに変換します。
- ④必要に応じて②～③を繰り返します。
- ⑤必要であればソースプログラムの本体のプログラムファイルエリアへの登録、印字を行います。

## 2. 2 機軸語エリア（ユーザーエリア）の確保

機軸語エリアは機軸語モニタのUSER命令で確保します。（275ページ参照）

確保しておかないとソースプログラムの変換などができません。

機軸語モニタモードにします。

**[BASIC] MON**

```
MACHINE LANGUAGE MONITOR
*
```

USER命令で機軸語エリアを確保します。

本機では、1Kワード（3KB）までのPIC用プログラムを作成することができます。

したがって、0CFFHまで（0100H～0CFFH）を確保しておきます。

**USER0CFF**

```
MACHINE LANGUAGE MONITOR
*USER0CFF
FREE:0100-0CFF
*
```

確保した機軸語エリアが表示されます。

## 3. ソースプログラムの作成・編集

本機のアセンブラでは、CASLのアSEMBルと同様、TEXTエリアに入れたソースプログラムをオブジェクトコードに変換します。（マイクロチップ・テクノロジー社のパソコン用開発ツール：MPLABに準拠）

ここでは、ソースプログラムを作成するための仕様（入力形式など）について説明します。

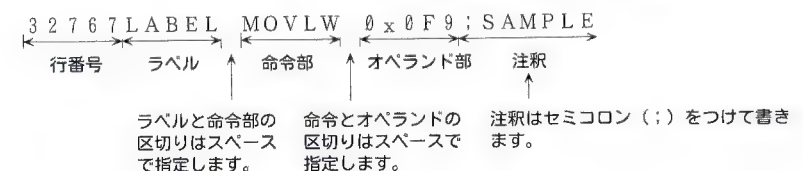
### 3. 1 ソースプログラムの構成

ソースプログラムの1行は通常、1ステートメント（1つの文）になります。

この行が何行か集まって、1つのプログラムが構成されます。

#### ◆行（ステートメント）の構成

1行は、行番号、ラベル、命令、オペランド、注釈、疑似命令などにより構成されます。



- 1行の長さは、注釈を含めて最大254文字までです。
- アルファベットの小文字は、オペランドで文字定数として書いた場合と、注釈に書いた場合以外は、大文字と同じものとみなします。

### ①行番号（ラインナンバー）

- 行番号は1～65279までの数値が使用できます。
- 1～65279の範囲外の値を指定すると“LINE NO. ERROR”と表示されます。

### ②ラベル

- ラベルは行番号に続けて入力します。（ラベルと行番号との間にスペースを入れると命令とみなされます。）
- ラベルは1～8文字まで使用できます。9文字以上入れるとエラーになります。
- ラベルには次の文字が使用できます。  
英文字（A～Z）、数字（0～9）、記号（\_）。  
ただし、数字はラベルの先頭には使えません（行番号と区別ができなくなります）。
- ラベルと、次の命令部の間は1つ以上のスペース（空白）で区切ります。`TAB`を用いることもできます。
- ラベルを書かないときは、行番号と次の命令の間に1つ以上のスペース（空白）を入れてください。`TAB`を用いることもできます。
- ラベルは最大102個まで定義できます。

### ③命令部（オペコード）

- 14ビット・コア用の35命令をニモニックコードで書き表します。また、311ページ以降に記載している疑似命令、`# INCLUDE`命令も用います。なお、命令をオペコード（オペレーションコード）と呼びます。
- 命令部と、次のオペランドとの間は1つ以上のスペース（空白）で区切ります。`TAB`を用いることもできます。

### ④オペランド部

命令（オペコード）に対して、その実行対象となるレジスタ、アドレス（番地）、定数などをオペランドといいます。

- オペランドが複数あるときは、コンマ（,）で区切ります。
- 定数には次のものが使用できます。

#### 【数値定数】

10進数、16進数が使用できます。

10進数……0～9の数字で表します。

〈例〉 1 8 8、3 2

16進数……先頭に0xをつけ、0～9、A、B、C、D、E、Fの16進用数字で表します。

〈例〉 0x B C、0x 2 0

#### 【文字定数】

文字は'（シングルクォーテーション）で囲んで表します。文字のアスキーコードが定数になります。

| 〈例〉 | （文字）     | （指定）  | （定数）   |
|-----|----------|-------|--------|
|     | A        | ' A ' | 0x 4 1 |
|     | ( NULL ) | ' '   | 0x 0   |

#### 【ラベル定数】

EQ U命令により、ラベルに定数を定義しておけば、そのラベルを定数として使用できます。

### ⑤注釈（コメント）

各行はセミコロン（;）をつけて注釈を書くことができます。セミコロン以降は、行の終わり（行末）まで注釈とみなされ、機械語（オブジェクト）には変換されません。

## 3.2 ソースプログラムの消去

TEXTモードの機能選択画面で、`DEL`を押せばデリート（Del）機能が選ばれ、次のようにテキスト内容を消去（削除）してよいか聞いてきます。（テキスト内容がない場合は、画面は変わりません。）

TEXT DELETE OK? (Y)

`Y`を押せば、テキスト内容がすべて消去され、TEXTモードの機能選択画面に戻ります。

くわしくは174ページを参照してください。

## 3.3 ソースプログラムの入力

TEXTモードの機能選択画面で`EDIT`を押せばエディット機能が選ばれます。

```
TEXT EDITOR
<
```

次にソースプログラムの入力手順を説明します。

- ①行番号を入力します。
- ②ラベルがないときは、`TAB`を押します。カーソルが命令部に移ります。  
`TAB`の代わりに`SPACE`でスペースを入力してもかまいません。  
スペースは1つ以上入力してください。  
ラベルを入力するときは、行番号に続けて入力します。この後は、`TAB`または`SPACE`でスペースを1つ以上入れます。
- ③命令を入力します。  
命令の後にオペランドがある場合は、`TAB`または`SPACE`でスペースを1つ以上入れます。
- ④オペランドを入力します。  
オペランドが複数ある場合は、コンマ（,）で区切って入力します。
- ⑤注釈をつける場合は、セミコロン（;）を入力して、その後に入力します。
- ⑥1行の入力を完了したら`ENTER`を押して、プログラムをメモリに格納します。`ESC`を押せばカーソルが消えます。

次の行を入力するときは、①から繰り返します。

## 4. アセンブラ

TEXTモードで入力したソースプログラムをPICマイコンが実行できるオブジェクトプログラムに変換する場合、まず、PICモードにします。

**[SHIFT] + [ASMBL]**を押してから**[P]**を押します。

メニュー画面になります。

**[A]**を押します。

アセンブルが実行されます。

アセンブル実行中は画面の下の方に“Assembling...”と表示され、終われば“Complete! (\*\*\*\*\* words)”と表示されます。アセンブルするプログラムが短いときは、この表示は一瞬で終わります。

(\*\*\*\*\*は、データ変換サイズ (word単位) を示します。)

### 4.1 アセンブラの疑似命令

疑似命令は、アセンブラを制御する命令で、それ自体はオブジェクトに変換されません。

本機のアセンブラには、次の疑似命令があります。

- コンフィグレーションビットの指定  
\_\_CONFIG命令
- プログラム開始番地の指定  
ORG命令
- ラベルの値の定義  
EQU命令
- データの定義  
DW命令

次に、これらの疑似命令の機能を説明します。

〈書式で使用している用語、記号の意味〉

式          数値、ラベル、'文字' を書くことができます。

[ ]          [ ] で囲まれた部分の省略ができることを示します。

#### \_\_CONFIG……コンフィグ

**機能** コンフィグレーションビットの設定をします。

**書式** \_\_CONFIG 式

**説明** ●各PICに従ったコンフィグレーションビットの設定をします。

コンフィグレーションビットの詳細は各PICの説明書を参照してください。

- MPASMでは“&” (ビットのAND) による指定ができますが、本機ではできません。

#### ORG……オリジン

**機能** プログラムの開始番地 (アドレス) を指定します。

**書式** ORG 式

**説明** ●ソースプログラムにORG命令を書かなかった場合は、ORG 0 の設定があるものとみなして、0番地からプログラムが開始されます。

- 式<sub>1</sub>は0Hから1FFFFHの範囲内で指定できます。ただし、オブジェクトプログラムがこの範囲を超えないように指定してください。(この範囲でも、メモリの制約によりエラーになる場合があります。)

〈例〉 ORG 0x0006

#### EQU……イー・キュー・ユー

**機能** ラベルにオペランドで指定した値を与えます。

**書式** ラベル EQU 式

**説明** ●ラベルに、オペランドで指定した値を定義付けます。式の演算はできません。

- 1または2バイトの値 (数値または'文字') を指定します。

〈例〉 START EQU 0x1000    ラベルSTARTに、0x1000が定義付けられます。STARTは定数0x1000として扱うことができます。

OK EQU 'Y'                ラベルOKに、0x59が定義付けられます。

#### DW……デファイン ワード

**機能** オペランドに記述した数値の下位2バイトをオブジェクトに生成します。

**書式** [ラベル:] DW 式

**説明** ●オペランドに記述された数値の下位2バイトをオブジェクトに生成します。

ただし、3FFFFH (14ビット) を超えた場合は14ビットでカットされます。

〈例〉 DW 0x1234          0x1234がオブジェクトになります。

### 4.2 #INCLUDE

**機能** 指定されたファイルをソースコードの一部として読み込み、標準的なラベルを使うことが可能になります。

**書式** #INCLUDE "ファイル名"

**説明** ●MPASMにある同名ファイルのラベル定義を、本機に保有しており使用できます。

ファイル名は" (ダブルクォーテーション) で囲んで表します。

なお、指定できるファイル名は次のとおりです。

P16F627.INC、P16F83.INC、P16F84.INC、P16F84A.INC

…14ビット・コアFlashメモリPIC

および

PIC, INC…上記以外

- 定義できるラベル数102個には含まれません。
- プログラムの先頭で指定します。
- MPASMで定義されている8文字を超えるラベルは、本機では次のとおり指定してください。

| MPASMで定義<br>されているラベル名 | 本機での指定   |
|-----------------------|----------|
| OPTION_REG            | OPTION_R |
| NOT_T1SYNC            | NOT_T1SY |

### 4.3 エラーメッセージ

アセンブラ機能を使用するときに、発生することがあるエラーのエラーメッセージと、その内容を記載します。エラーは[CLS]で解除できます。また、[BASIC]や[TEXT]などでモードを切り替えた場合も解除されます。

| エラーメッセージ                | 内 容 (原 因)                                                                                                                                                                                                                                                                                      |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| File not exist!         | TEXTエリアにアSEMBルするプログラムが入っていない(存在しない)。                                                                                                                                                                                                                                                           |
| No USER AREA!           | 機械語エリアが確保されていない。<br>ラベルエリア分の機械語エリアが確保されていない。                                                                                                                                                                                                                                                   |
| Not __CONFIG data!      | __CONFIG疑似命令がない。                                                                                                                                                                                                                                                                               |
| Syntax error! (****)    | __CONFIGの前にラベルがある。<br>ORG指定の前にラベルがある。<br>EQU指定の前にラベルがない。<br>不当ニモニック、疑似命令がある。<br>ニモニック、疑似命令のオペランドの後ろに、SPACE/TAB/CR/; 以外がある。<br>オペランドなしの命令の後ろに、SPACE/TAB/CR/; 以外がある。<br>オペランド1と2の間に、SPACE/TAB/, 以外がある。<br>オペランド1と2の間に、, がない。<br>不当なオペランドがある。<br>不当なプリプロセッサコマンドがある。<br>プリプロセッサコマンドのフォーマットに間違いがある。 |
| Out of range! (****)    | 結果格納先指名子が1ビット以上ある。<br>リテラル値が8ビット以上ある。<br>ビット指定値が3ビット以上ある。<br>オペランド値等が16ビット以上ある。                                                                                                                                                                                                                |
| Undefined label! (****) | オペランドで使用しているラベルがない。                                                                                                                                                                                                                                                                            |
| Undefined line! (****)  | アドレス値が機械語エリアを超えている。                                                                                                                                                                                                                                                                            |
| Label too long! (****)  | ラベルが8文字以上ある。                                                                                                                                                                                                                                                                                   |
| Out of memory! (****)   | ORG指定が8K以上を指している。<br>オペコードが機械語エリアを超えている。<br>ラベル数がMaxを超えている。                                                                                                                                                                                                                                    |

| エラーメッセージ                  | 内 容 (原 因)                     |
|---------------------------|-------------------------------|
| Multi define! (****)      | 同一ラベルが複数ある。<br>#includeが複数ある。 |
| 'Not include file! (****) | 指定includeファイル名が不当。            |

(\*\*\*\* は、エラー発生行番号を示します。)



# 第10章

## BASICの各命令の説明

この章ではBASICの各命令を1つ1つ説明します。

本機がどのような命令をもっているかを見て頂いた後で、必要な命令の説明をお読みください。

以降の説明で書式などに使用する用語の意味を示します。

式……………数値、数値変数およびこれらを含んだ計算式を示します。

変数……………配列要素を含んだ数値変数、文字変数を示します。

'文字' ……' ' (ダブルクォーテーションマーク) で囲まれたキャラクタ (文字、数字、記号) を示します。

文字列……………'文字'、文字変数を示します。

( ) ……カッコでくくる必要があることを示します。

[ ] ……省略可能であることを示します。ただし、この後にほかの指定を行う場合は、コンマなどの区切り記号が必要です。

{A  
B} ……AあるいはBを選択することができます。

プログラム ……プログラムによる実行が可能です。

マニュアル ……マニュアル操作による実行が可能です。

省略形……………命令の中には、そのつづりを省略した形で入力できるものがあり、その場合は最も省略した形で示しています。

〈例〉 省略形…P. これはPRINTの省略形ですが、次の形でも有効になります。

PR.

PR I.

PR I N.

(注) ・変数などに続いてBASIC命令を入力する場合は、変数と命令の間にスペースを入力してください。

〈例〉 5 0 I F A=B THEN 1 0 0  
                                  ↑  
                                  スペースを入れる。

・関数命令などをマニュアルで実行する場合はRUNモードで実行してください。

PROモードで実行するとエラー12になります。

ただし、PRINT命令の中で使用すれば、PROモードでも使用できます。

〈例〉 PRINT CHR\$ 9 0 → Z

### 関数

AND……………アンド

プログラム

省略形……………AN.

マニュアル

機能

式と式との論理積を計算します。また、条件式の結合を行います。

書式

式 AND 式

条件式 AND 条件式

参照

OR、NOT、IF

説明

● 2進数において、論理積は次のような値を取ります。

1 AND 1 = 1      0 AND 1 = 0

1 AND 0 = 0      0 AND 0 = 0

10進数の論理積を求めた場合は、計算機内では10進数を2進数に変換してから各桁の論理積を求め、その結果を10進数に戻します。

たとえば、41と27の論理積は次のように計算されます。

4 1 AND 2 7 = 9

AND  $\begin{array}{r} 101001 \cdots 41 \\ 011011 \cdots 27 \\ \hline 001001 \cdots 9 \end{array}$

41と27をそれぞれ2進数に変換し、各桁のANDを取ります。  
そして、その結果を10進数に変換すれば9になります。

● 式の値は-32768～32767の整数部が有効になります。

● 2つ以上の条件をすべて満足するような条件を1つの式で表します。

〈例〉 5 0 I F B > 5 AND C > = 4 THEN ……

もし、Bが5よりも大きく、かつCが4よりも大きいとき、  
等しいとき、THENに続く命令を実行します。

### 関数

ASC……………アスキー

プログラム

省略形……………AS.

マニュアル

機能

文字や記号、数字などをキャラクタコードに変換します。

書式

ASC { "文字"  
文字変数 }

参照

CHRS

説明

● 文字や記号、数字などをキャラクタコードに変換します。たとえば「Z」という文字のキャラクタコードを知りたい場合は

A = ASC ' Z '

として実行すれば、Aには「Z」のキャラクタコードが10進数の90として代入されます。

文字が2文字以上指定された場合は、先頭の文字のみがキャラクタコードに変換されます。

〈例〉 1 0 CLS

2 0 AS = INKEY\$

```

30 IF AS$ = " " THEN 20      プログラムをスタートさせた後、アルファ
40 B=ASC A$                  ベットや数字キーなどを押せば、その文字
50 PRINT A$; " "; B          のキャラクタコードを表示します。

```

- 文字、数字、記号など（これらを総称してキャラクタといいます）を計算機が記憶したり、処理したりする場合は、すべて計算機が取り扱いやすい数値に変換します。たとえばアルファベットのAは計算機内では65（10進数）という数値（コード）になっています。（実際には2進数の01000001となっています。）同様にBは66、Cは67というようにコードを決めています。このコードの決めかたに何種類かあって、代表的なものにアスキーコード（ASCII code）とJISコードがあります。

本機ではJISコードを元にして作成されたキャラクタとそのコードを使用しています。（384ページのキャラクタ・コード表を参照してください。）


## 基本命令

### AUTO……オート




省略形……なし



マニュアル

**機能** 行番号の自動発生を行います。（PROモードのマニュアル操作でのみ有効）

**書式** AUTO [[開始行] [, 増分]] 





**参照** NEW、PASS


- 説明**
- 最初に開始行で指定された行番号を発生して、入力待ちになります。1行分のプログラムを入力して  を押すと、増分で指定した値を加えた行番号を発生して入力待ちになります。以下、同様に  を押すたびに新たな行番号を発生します。
  - 開始行や増分に0、負数、65279を超える値を指定したときはエラーになります。
  - 開始行や増分を省略したときは、それぞれ10が指定されます。ただし、前回AUTO命令で指定された値があれば、その値が指定されます。
  - PROモードからRUNモードに移り、再びPROモードに戻ったとき、AUTO  と操作すると、前の設定値が有効で、引き続いた行番号を発生します。

また、行番号を表示しているときに  や  を押したときも同じです。

- パスワードが設定されているときは、AUTO命令は無視されます。

- 次の場合は設定値（開始行、増分値）が解除されます。

-  +  と操作したとき
- 電源をオフ、オンしたとき
- LOAD、NEW命令を実行したとき
- BLOAD命令を実行し、BASICプログラムが読み込まれたとき
- FILES命令により一覧されたファイルから選択後、 +  の操作をしたとき
- BASICコンバータでBASICへの変換（Basic ← Text）を実行したとき

〈例〉 AUTO 100, 10 

開始行番号を100とし、110、120、……と行番号を発生します。

## ポケコン間通信命令

### BLOAD……バイナリ・ロード

省略形……BLO.

マニュアル

**機能** 別のポケコンから本機にBASICプログラムを読み込ませます。（PROおよびRUNモードのマニュアル操作でのみ有効）

**書式** BLOAD 

**参照** BLOAD?、BSAVE

**説明**


- 別のポケコンに記録されているBASICプログラムを本機に読み込みます。

- 読み込みが行われているときは、画面右下に\*マークが表示されます。読み込みが終われば、\*マークは消えます。

BASICプログラムの受信待ちのときは、まだ読み込みが行われていけませんので\*マークは表示されません。

これは、BLOAD?命令でも同じです。

なお、読み込みが終われば、プロンプト記号">"が表示されます。

- 別のポケコンにBASICプログラムがなかった場合でも、本機はBASICプログラムを検索し続けます。この場合は  を押して検索を止めてください。これは、BLOAD?命令でも同じです。

- 実行中にエラーが発生すると、計算機内のBASICプログラムが無効になります。

- 実行するとオープンされていた全ファイルが閉じられます。（クローズされます。）

## ポケコン間通信命令

### BLOAD M…バイナリ・ロード・エム

省略形……BLO. M

マニュアル

**機能** 別のポケコンから機械語プログラムを読み込みます。

**書式** BLOAD M [ロード番地] 

**参照** BLOAD、BSAVE M

**説明**

- BSAVE M命令で別のポケコンに書き込んだ機械語プログラムを読み込みます。そのとき、本体のプログラムは消去しません。

- 「ロード番地」を指定したときは、指定した番地を開始番地として読み込みます。省略すると、BSAVE M命令で指定した番地に読み込みます。

## ポケコン間通信命令

### BLOAD?…バイナリ・ロード?

省略形……BLO. ?

マニュアル

**機能** 本機内のBASICプログラムと別のポケコンに記録されている内容との照合を行います。（PROおよびRUNモードのマニュアル操作でのみ有効）

**書式** BLOAD? 

**参照** BLOAD、BSAVE

**説明**

- 照合は、BASICプログラムが正しく別のポケコンに記録されたか、あるいは別のポケコンから正しく読み込まれたか確認するために行います。

- 照合において、もし内容に不一致が生じたときはエラー82になります。

- BASICプログラムの照合が行われているときは画面右下に\*マークが表示されます。照合が終われば\*マークは消え、プロンプト表示">"になります。

## ポケコン間通信命令

## BSAVE……バイナリ・セーブ

プログラム

## 省略形……BS.

マニュアル

- 機能** BASICプログラムを別のポケコンに記録します。
- 書式** BSAVE
- 参照** PASS、BLOAD、BLOAD?
- 説明**
- 別のポケコンに記録します。
  - パスワードが設定されているときは、BSAVE命令は無視されます。
  - 実行するとオープンしていた全ファイルが閉じられます。

## ポケコン間通信命令

## BSAVE M…バイナリ・セーブ・エム

プログラム

## 省略形……BS. M

マニュアル

- 機能** 機械語プログラムを別のポケコンに記録します。
- 書式** BSAVE M 開始番地, 終了番地
- 参照** BSAVE、BLOAD M
- 説明**
- バイナリ形式で別のポケコンに記録します。
  - 指定した開始番地から終了番地までの内容を書き込みます。
  - 番地は10進数、または&Hをつけて16進数で指定します。

## 基本命令

## CALL……コール

プログラム

## 省略形……CA.

マニュアル

- 機能** 機械語プログラムを実行します。
- 書式** CALL 番地
- 参照** PEEK、POKE
- 説明**
- 指定された番地から機械語プログラムを実行します。
- CALL &H1F58
- (注) この命令を誤って使用するとBASICプログラムや本機のシステムエリアを破壊し、異常が発生することがあります。この場合はリセットスイッチを押して、メモリの内容を消去してください。

## 関数

## CHR\$……キャラクタドル

プログラム

## 省略形……CH.

マニュアル

- 機能** キャラクタコードを文字や記号など（キャラクタ）に変換します。
- 書式** CHR\$ 式
- 参照** ASC
- 説明**
- CHR\$はASC命令とは逆の関数で、キャラクタコードを文字や記号、数字に変換します。たとえば、キャラクタコード「90」の文字を知りたい場合は  
 $AS = CHR\$ 90$   
 として実行すれば、A\$には「Z」が代入されます。
  - 本機で扱うことのできるキャラクタと、それに対応するコードについては384ページのキャラクタ・コード表を参照してください。
- 〈例〉
- ```

10 AA$ = ''
20 INPUT "コード" = ; A : CLS
30 AA$ = AA$ + CHR$ A
40 LOCATE 7, 1 : PRINT AA$
50 GOTO 20

```
- このプログラムは、キャラクタコードを入力し、それを文字や記号に変換して文字変数A\$に順次格納していきます。変換されたキャラクタはそのつど表示されます。プログラムを実行させ、次のコードを入力してみてください。
- 234、83、72、65、82、80（“◆SHARP”が表示されます）
- 256以上の値が指定されたときは、エラー33になります。

## グラフィック命令

## CIRCLE……サークル

プログラム

## 省略形……CI.

マニュアル

- 機能** DEGモードで円、扇形や楕円などを描きます。
- 書式** CIRCLE (式<sub>1</sub>, 式<sub>2</sub>), 式<sub>3</sub> [, [式<sub>4</sub>], [式<sub>5</sub>], [式<sub>6</sub>], [式<sub>7</sub>]]
- 参照** GCURSOR、LINE、PSET
- 説明**
- (式<sub>1</sub>, 式<sub>2</sub>)で指定した点を中心として、式<sub>3</sub>で指定した長さ（ドット数）を半径とする円を描きます。

(0, 0)

式<sub>1</sub> : 中心点のX座標 画面左端は0、右端は143です。  
 式<sub>2</sub> : 中心点のY座標 画面上端は0、下端は47です。



(143, 47)

- 式<sub>1</sub>、式<sub>2</sub>の値は-32768～32767の範囲内で指定できますが、上記の範囲を超えた場合は、画面外の指定の点を中心として円を描きます。
- 式<sub>3</sub>で半径をドット数で指定します。式<sub>3</sub>の範囲は1～32767です。

- 式<sub>4</sub>で開始角度、式<sub>5</sub>で終了角度を度単位で指定します。  
式<sub>4</sub>、式<sub>5</sub>は0～360度の範囲内で指定します。中心座標の右側が0で反時計方向に描きます。  
省略した場合は、開始角度は0、終了角度は360が指定されます。(円になります。)
- 角度指定はDEGモードにしてください。RAD、GRADモードでは正しく描かれせん。
- 開始角度、終了角度に負の値を指定したときは、円周から中心点までの線(半径線)を描きますので、扇形を描くことができます。負の値に対してはその絶対値が指定されたものとして描きます。  
なお、0°の半径線を描くときで角度を変数で指定するときは、-360°と指定するか、もしくは-1\*変数名で指定してください。変数に-0を代入しても+0として認識されます。
- 式<sub>6</sub>の値により比率を指定します。  
比率が1のときは円で、それ以外のときは楕円を描きます。省略した場合は1が指定されます。

$$\text{比率} = \frac{\text{Y軸方向の半径}(r_y)}{\text{X軸方向の半径}(r_x)}$$

比率を0.5にすると横長の楕円になります。

- S、R、Xで円周に相当するドットを点灯させるか、消すか、あるいは反転させるかを指定します。省略した場合はSが指定されます。  
S…円を描くとき、ドットを点灯させて円を描きます。(ドットをセット)  
R…円を描くとき、ドットを消灯させて円を描きます。(ドットをリセット)  
円の回りのドットが点灯している場合に円を描くときや、描かれている円を消すときなどに用います。  
X…円を描くとき、円に相当するドットが点灯しているときは消して、消えているときは点灯させます。(ドットの反転)
- 式<sub>7</sub>で円の内部の模様を次のように指定します。省略した場合は0が指定されます。



〈例〉 1 0 CLS  
2 0 CIRCLE (71, 23), 20, , , 0.5, , 2  
3 0 END



横長の楕円を描きます

1 0 CLS  
2 0 CIRCLE (71, 23), 20, -45, -135  
3 0 END



扇形を描きます

- (注) ● 円の一部を描く場合でも、円のすべてが範囲内(-32768～32767)にないとエラー33になります。
- 画面はドットで構成されていますので、円、斜線、曲線などは正確な線にならない場合があります。
  - 半径が小さい扇形を描くとき、描画精度が悪くなる場合があります。
  - CIRCLE命令を実行するためには、ワーク用として約1440バイト以上のフリーエリアが必要です。

## 基本命令

### CLEAR……クリア

プログラム

### 省略形……CL.

マニュアル

- |    |                                                                                                                                                                                                                           |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 機能 | 固定変数の内容および配列変数、単純変数を消去します。                                                                                                                                                                                                |
| 書式 | CLEAR                                                                                                                                                                                                                     |
| 参照 | DIM                                                                                                                                                                                                                       |
| 説明 | <ul style="list-style-type: none"> <li>● 単純変数や、DIM命令により確保されていた配列変数はすべて消去(未定義の状態に)され、固定変数の内容もすべて消されます。</li> </ul> <p>〈例〉 1 0 0 CLEAR: DIM B(4)</p> <p>この例では、すべての変数を消去してから、配列変数B(4)の定義を行っています。プログラムの先頭ではよくこのような方法を用います。</p> |

## ファイル関連命令

### CLOSE……クローズ

プログラム

### 省略形……CLOS.

マニュアル

- |    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 機能 | 入出力のファイルを閉じます。                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 書式 | CLOSE [#ファイル番号, #ファイル番号……]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 参照 | OPEN、PRINT#、INPUT#                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 説明 | <ul style="list-style-type: none"> <li>● ファイル番号に対応するファイルを閉じます。ファイル番号は1、2または3のいずれかです。</li> <li>● ファイル番号を省略するとすべてのファイルを閉じます。</li> <li>● ファイルが出力(OUTPUTまたはAPPEND)モードでオープンしていた場合は、メモリ(出力バッファ)に残っているデータおよびファイル終了コードを、出力してからクローズします(閉じます)。</li> <li>● 次の場合も自動的にファイルを閉じます。 <ul style="list-style-type: none"> <li>・END、NEW、RUN命令を実行したとき</li> <li>・電源が切れたとき</li> <li>・プログラムの編集をしたとき(プログラムの入力、修正、削除やDELETE命令などを実行したとき)</li> <li>・プログラムファイルエリアへのプログラムの登録、呼び出し、削除などを行ったとき</li> <li>・BSAVE、BLOAD、BLOAD?、BSAVE M、BLOAD M命令実行時</li> <li>・機械語モニタモードへ切り替えたととき</li> <li>・BASICモード(RUN、PRO)から他のモード(TEXT、CASLなど)へ切り替えたととき</li> </ul> </li> </ul> |



## 基本命令

CLS……………クリアスクリーン

プログラム

省略形…………なし



|    |                               |
|----|-------------------------------|
| 機能 | 表示内容を消去します。                   |
| 書式 | CLS                           |
| 参照 | LOCATE                        |
| 説明 | 表示内容を消去し、表示開始位置を（0，0）位置に戻します。 |

## 基本命令

CONT……………コンティニュー

マニュアル

省略形…………C.

|    |                                                                                                                          |
|----|--------------------------------------------------------------------------------------------------------------------------|
| 機能 | 一時停止しているプログラムの実行を再開します。（RUNモードのマニュアル操作でのみ有効）                                                                             |
| 書式 | CONT                                    |
| 参照 | STOP                                                                                                                     |
| 説明 | STOP命令や  によりプログラムが一時停止しているとき、実行を再開させます。 |

## 基本命令

DATA……………データ

プログラム

省略形……………DA.

|    |                                          |
|----|------------------------------------------|
| 機能 | READ文に続く変数に与えるデータを指定します。                 |
| 書式 | DATA { 式 }, { 式 } ……<br>{ 文字列 }, { 文字列 } |
| 参照 | READ                                     |
| 説明 | ● READ命令の説明を参照ください。                      |

## 基本命令

DEGREE…ディグリー

プログラム

省略形……………DE.

マニュアル



|    |                                                        |
|----|--------------------------------------------------------|
| 機能 | 角度単位を“度”に設定します。                                        |
| 書式 | DEGREE                                                 |
| 参照 | RADIAN, GRAD                                           |
| 説明 | ● 三角関数、逆三角関数、座標変換で扱う角度の単位を“度”単位〔°〕に設定します。（1直<br>角=90°） |

## 基本命令

DELETE…デリート

マニュアル

省略形……………DEL.

|    |                                                                                                                                                                                                                                                                                                                                        |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 機能 | プログラム行を削除します。（PROモードのマニュアル操作でのみ有効）                                                                                                                                                                                                                                                                                                     |
| 書式 | （1）DELETE 開始行番号 [— [終了行番号]] <br>（2）DELETE —終了行番号                                                                                                                 |
| 参照 | NEW, PASS                                                                                                                                                                                                                                                                                                                              |
| 説明 | ● 開始行番号と終了行番号を指定したときは、その間に含まれるすべての行を削除します。<br>● 開始行番号のみを指定したときは、その行だけを削除します。<br>● 開始行番号とハイフン（—）を指定したときは、開始行番号以降のすべての行を削除します。<br>● 書式（2）では、プログラムの先頭行から終了行番号間に含まれるすべての行を削除します。<br>● 開始行番号と終了行番号の両方を省略したときは、エラー10になります。<br>● 指定した行番号が存在しないときは、エラー40になります。<br>● 開始行番号が終了行番号よりも大きい指定をすると、エラー44になります。<br>● パスワードが設定されているときは、DELETE命令は無視されます。 |

## 基本命令

DIM……………ディメンジョン

プログラム

省略形……………D.

マニュアル

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 機能 | 配列名と、その大きさを定義（宣言）し、配列変数をメモリ（プログラム・データエリア）上に確保します。                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 書式 | （1）DIM { 配列名（式 <sub>1</sub> ） } [ , { 配列名（式 <sub>1</sub> ，式 <sub>2</sub> ） } … ]<br>（2）DIM { 配列名（式 <sub>1</sub> ） } *式 <sub>3</sub> [ , { 配列名（式 <sub>1</sub> ，式 <sub>2</sub> ） } *式 <sub>3</sub> … ]<br>（注）書式（2）は文字変数でのみ使用できます。                                                                                                                                                                                                                                            |
| 参照 | CLEAR, RUN, ERASE                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 説明 | ● 配列変数を使用するときは、事前にDIM命令により配列名と、その大きさを定義（宣言）して、メモリ（プログラム・データエリア）上に確保しておく必要があります。<br>● 配列名はアルファベット1文字あるいは2文字（2文字目は数字も使用可能）で指定し、文字配列変数の場合は後に\$マークをつけます。<br>● 式 <sub>1</sub> および式 <sub>2</sub> は添字といわれ、配列の大きさ（配列要素数）と次元を指定します。添字が1つのものを一次元配列と呼び、2つのものを二次元配列と呼びます。本機では二次元配列までが使用できます。<br>〈例〉 DIM B (3)                   一次元配列変数B（ ）について、配列要素B(0)、B(1)、B(2)、B(3)の4個が確保されます。<br>DIM XAS (2, 3)       二次元配列変数XAS（ ）について、配列要素XAS(0, 0)、XAS(0, 1)、……XAS(2, 2)、XAS(2, 3)の12個が確保されます。 |

添字は理論的に0～255までの整数値を用いることができますが、計算機のメモリの大きさ、使用状態によっては添字で指定しただけ、変数が確保できない場合があります。（確保できないときはエラー60になります。）

- 添字が小数部を含んでいるときは小数部は無視され、整数部のみが有効になります。
- 添字は数値変数や式の形で用いることもできます。

〈例〉 10 INPUT A, B  
      20 DIM X (A), Y (B-1)

- 文字配列変数は変数の長さを指定できます。

書式(2)において、式<sub>3</sub>により文字数を1～255文字の範囲で任意に指定できます。

〈例〉 DIM F\$(2)\*30 F\$(0)～F\$(2)の各変数には、それぞれ最大30文字まで記憶できます。

DIM Y\$(5,4)\*6 Y\$(0,0)～Y\$(5,4)の各変数には、それぞれ最大6文字まで記憶できます。

文字数の指定(\*式<sub>3</sub>)を省略した場合は、自動的に16文字が指定されます。

指定文字数が大きいほど多くのメモリを必要とします。

- 複数の配列を使用する場合は、DIM命令で一度に定義することができます。

〈例〉 DIM J(5), K\$(4,3), XB\$(5)\*10

- 一度定義した配列名は再定義できません。

たとえば、DIM X(5)とDIM X(3,4)は同じXという配列名になり、同時に使用できません。ただし、数値配列変数と文字配列変数は別の配列とみなされます。たとえば、配列Z( )とZ\$( )は同時に使用できます。

- 配列変数はERASE、CLEAR命令により消去する(未定義の状態にする)ことができます。また、RUN命令によりプログラムの実行を開始したときも、以前に定義されていた配列変数はすべて消去されます。GOTO命令によるプログラムの実行では、変数は消去されません。したがって、一度実行したプログラムをGOTO命令により再実行させるときなどに、DIM命令の書かれている行を実行させると同じ変数名を再定義することになり、エラー30になります。このような場合はERASEまたはCLEAR命令で消去してから、定義し直すようにしてください。

〈例〉 50 'S':ERASE X: DIM X(3,4)

## 基本命令

### DMS\$……ディー・エム・エス・ドル

プログラム

#### 省略形……DM.

マニュアル

**機能** 10進数(度)を60進数(度・分・秒)の文字列に変換します。

**書式** DMS\$ 式

**参照** VDEG

**説明** ●10進数を60進数の文字列(度(°)・分(')・秒(")の記号を含む)の形に変換します。

〈例〉 10 B=6/1.01  
20 A\$=DMS\$ B  
30 PRINT A\$

```
RUN
61° 00' 36"
>
```

(注) 変換した結果を格納する文字変数には、文字単純変数か文字配列変数を使用してください。文字固定変数には、最大7文字まで格納できます。(変数の長さについては153ページを参照)

〈例〉 10 B=1.2345  
20 A\$=DMSS\$ B  
30 PRINT 'A\$=' ; A\$  
40 AA\$=DMS\$ B  
50 PRINT 'AA\$=' ; AA\$

```
RUN
A$= 1° 14' 04
AA$=1° 14' 04. 2"
>
```

- 度・分・秒の記号のキャラクターコード(くわしくは384ページ参照)  
度(°) …223 (H&DF)、分(') …39 (H&27)、秒(") …248 (H&F8)

## 基本命令

### END………エンド

プログラム

#### 省略形……E.

**機能** プログラムの実行を終了させます。

**書式** END

**説明** ●プログラムの実行を終了し、オープンされていたファイルをすべて閉じます。

## 基本命令

### EOF………エンド・オブ・ファイル

プログラム

マニュアル

#### 省略形……EO.

**機能** ファイルの終わりを検出します。

**書式** EOF (ファイル番号)

**参照** OPEN

**説明** ●ファイル番号で指定したファイルのデータを最後まで読み取ったかどうかを調べます。ファイル番号は1、2、3のいずれかです。

- ラムデータファイルまたは11ピンの通信時のみ有効です。
- 最後に達していれば-1(真)、そうでなければ0(偽)の値を与えます。
- 11ピン通信時で全二重通信の場合は、受信バッファ内が空のときは-1(真)、空でないときは0(偽)になります。
- 半二重通信の場合は、最後に達していれば-1(真)、そうでなければ0(偽)になります。
- ファイル番号で指定したファイルの装置名がE(ラムデータファイル)の場合は、INPUTモードでオープンされていないとエラーになります。

〈例〉 ラムデータファイル(ABC.DAT)の確保(181ページ参照)をした後で、次のプログラムを入力してください。

```
10 OPEN 'E:ABC.DAT' FOR OUTPUT AS #2
20 PRINT #2, 123, 456, 789
30 CLOSE
40 OPEN 'E:ABC.DAT' FOR INPUT AS #2
50 INPUT #2, A, B
60 X=EOF(2) ← データの読み出しは終了していない
70 INPUT #2, C
80 Y=EOF(2) ← データの読み出しは終了している
90 CLOSE :PRINT X, Y:END
```

このプログラムを実行すると、X=0、Y=-1になります。

## 基本命令

### ERASE………イレーズ

プログラム

マニュアル

#### 省略形……ER.

**機能** 配列変数を消去します。

**書式** ERASE 配列名 [, 配列名……]

参照

CLEAR、DIM

説明

- 指定した配列変数を消去します。

- ERASE命令で配列変数を消去すれば、その配列変数が占めていた領域を他の目的のために使うことができます。

また、配列の大きさを変更したいとき、ERASE命令でそれまでの配列を消去すればDIM命令で配列の大きさを再定義できます。

- すべての変数を消去するときはCLEAR命令を使います。

```

<例>  1 0  DIM A (2, 3), B$ (5) * 3 0
        :
        1 0 0  ERASE B$
  
```

## ファイル関連命令

### FILES……ファイルズ

省略形……F I .

マニュアル

機能

プログラムファイルエリアに登録されているファイルのファイル名とファイルサイズを表示します。

書式

FILES [↓]

参照

LFILES、SAVE、LOAD

説明

- 実行すると先頭のファイル名から表示します。画面には6つのファイル名まで表示されます。
- ファイル名の左側に➡マークが点灯しています。➡マークは(▲)、(▼)で上下に移動できます。これを移動させていけば必要に応じて画面が送られて、別のファイル名を表示します。  
➡マークを必要なプログラムのファイル名に移動させてから(SHIFT) + (M)を押すと、そのプログラムを呼び出すことができます。ただし、TEXTモードで登録したプログラムを呼び出すことはできません。
- ファイルサイズについてのくわしい説明は、179ページをご覧ください。
- (CLS)、(BREAK)などでファイル名の表示を解除できます。

## 基本命令

### FIX……フィックス

省略形……なし

プログラム

マニュアル

機能

数値の整数部を求めます。

書式

FIX 式

説明

- 式の値の小数点以下を取り除いた整数部だけの値を求めます。

```

<例>  A=FIX  2.5           Aには2が代入されます。
        A=FIX  -2.5         Aには-2が代入されます。
        A=FIX (2.45*3)      Aには7が代入されます。
  
```

## 基本命令

### FOR～NEXT…フォー～ネクスト

省略形……F. N. STE.

プログラム

機能

FORとNEXTの間に書かれた命令を指定された条件が満たされるまで、繰り返し実行します。

書式

```

FOR 数値変数 初期値 TO 最終値 STEP きざみ値
    }
NEXT [数値変数]
  
```

参照

REPEAT～UNTIL、WHILE～WEND

説明

- 数値変数が初期値から始まって、指定されたきざみ値分ずつ増加（あるいは減少）していき、最終値よりも大きく（あるいは小さく）なるまでFORとNEXTの間を繰り返し実行します。（この繰り返し部分をFOR～NEXTループと呼びます。）

```

<例>  FOR A=0 TO 10 STEP 2  Aが0から始まって、1回FORとNEXT
                                T間を実行するごとにAに2を加えながら、
                                Aの値が10を超えるまで、FORとNEXT
                                の間を実行します。
        }
        NEXT A
  
```

- きざみ値が1のときはSTEP 1を省略することができます。
- FORとNEXTは必ず対にして使い、FORの後の数値変数とNEXTの後の数値変数は同一でなければなりません。ただし、NEXTの後の数値変数は省略できます。

```

FOR B=1 TO 5
    }
NEXT B
  
```

← 同じ数値変数にする。

- 初期値、最終値、きざみ値（ステップ値）は次の範囲内で指定できます。  
-9.999999999E99～9.999999999E99  
(-9.999999999×10<sup>99</sup>～9.999999999×10<sup>99</sup>)
- 初期値にきざみ値を加えると最終値から離れてしまう場合は、ループ内を1回だけ実行してループを抜けます。  
なお、きざみ値を0に指定しますと、永遠にループ内の実行を繰り返すプログラムになってしまいます。
- FOR～NEXTループの中に、別のFOR～NEXTループを入れる場合、中に入るFOR～NEXTループは外のFOR～NEXTループ内に完全に入っていなければなりません。この条件でループを最大5段まで重ねて使う（深みをもたせる）ことができます。（379ページのスタック参照）
- FOR～NEXTループの外からループ内に飛び込むことはできません。（飛び込ませるとエラー52になります。）  
(注) ・FOR～NEXTループから外に飛び出した場合、そのループは終了したことになります。プログラムによっては（FOR命令を何回か実行するようなプログラムの場合）、FOR～NEXTの深みエラー50が発生することがあります。  
・FOR～NEXTループ内ではCLEAR、ERASE、DIM命令は使用できません。

## 関数

### FRE……フリー

省略形……F R .

プログラム

マニュアル

機能

未使用部分のバイト数を求めます。

書式

FRE

説明

- 本機内のメモリの中で、現在使用されていない部分のバイト数が得られます。  
BASICのプログラムや配列変数、単純変数および機械語エリア、プログラムファイルエリア、ラムデータファイルエリア、テキストエリアとして使用されている部分以外のバイト数を求めます。

## グラフィック命令

## G CURSOR.....グラフィックカーソル

プログラム

## 省略形.....GC.

マニュアル

機能

グラフィック表示の開始位置をドット（点）単位で指定します。

書式

G CURSOR (式<sub>1</sub>, 式<sub>2</sub>)

参照

G PRINT、LOCATE

説明

- 画面は横144、縦48のドット（点）で構成されています。

このドットはそれぞれ横方向に0～143、縦方向に0～47の番号がつけられています。この番号をX-Y座標と同様の形、つまり式<sub>1</sub>でX方向、式<sub>2</sub>でY方向の番号を指定して、表示開始位置（表示の開始ドット）を指定します。

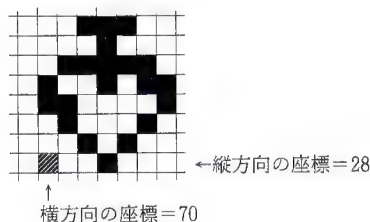


- 式<sub>1</sub>、式<sub>2</sub>は-32768～32767の範囲で指定できます。ただし、画面をはみ出すような指定を行う（式<sub>1</sub>が0～143、式<sub>2</sub>が0～47の範囲外の値）と、画面外の仮想位置を表示開始位置に指定することになります。
- 表示開始位置はRUN命令を実行したときや電源を入れ直したとき、または、(SHIFT) + (CA)を押したときなどでは(0, 7)位置が指定されます。

〈例〉 10 CLS : WAIT  
20 G CURSOR (70, 28)  
30 G PRINT "183458F452C18"

このプログラムを実行すると、画面の中程に次のように表示されます。

（斜線部分は表示されません。）



## 基本命令

## G OSUB~RETURN...ゴースブ~リターン

プログラム

## 省略形.....G OS. RE.

機能

指定した行から始まるサブルーチンへプログラムの実行を移し、RETURN命令で戻ります。

書式

G OSUB { 行番号  
ラベル }

RETURN

参照

GOTO、ON GOSUB

説明

- 何回も同じ計算や処理が出てくる場合、その部分を抜き出してプログラムしておきます。抜き出したプログラムを必要に応じて実行すれば、プログラムを短かく簡略化できます。
- サブルーチンへのジャンプはGOSUB命令に続いて、サブルーチンのおかれている行番号またはラベルを書いて指示します。（ラベルについては163ページ参照）

〈例〉  
:  
50 : GOSUB 200 200行へサブルーチンジャンプ  
:  
100 : GOSUB "A" ラベル "A" または \* A が書かれている行へ  
:  
サブルーチンジャンプ

- サブルーチンの最後にはRETURN命令を書いて、メインルーチンへの復帰を指示します。（RETURN命令には行番号の指定は不要です。）  
メインルーチンへ復帰したときは、GOSUB命令の次の命令を引き続き実行します。
- サブルーチンから別のサブルーチンへ実行を移すことができます。  
サブルーチンからサブルーチンへ、また次のサブルーチンへ……というように重ねて使用する場合、最高10段まで重ねる（深みをもたせる）ことができます。（10段を超えるとエラー50になります。）

## 基本命令

## GOTO.....ゴートゥー

プログラム

## 省略形.....G.

マニュアル

機能

プログラムの実行を指定された行へ無条件に移します。

書式

GOTO { 行番号  
ラベル }

参照

GOSUB、ON~ GOTO、RUN

説明

- 通常プログラムは小さい行から順次実行されますが、GOTO命令により、その実行を指定した行へ移す（ジャンプさせる）ことができます。  
また、RUNモードでのマニュアル操作により、指定した行からプログラムの実行を開始させることができます。
- ジャンプ先は、GOTO命令に続けて行番号またはラベルを書いて指定します。（ラベルについては163ページ参照）  
〈例〉 GOTO 40 40行へジャンプしない。  
GOTO "AB" ラベル "AB" または \* AB のついている行へジャンプしない。
- 指定した行番号およびラベルがない場合はエラー40になります。
- 同じラベルが2個以上書かれているときは、行番号の小さいほうへジャンプします。
- GOTO命令によりプログラムの実行が開始されたときの計算機の状態については163ページを参照してください。

## グラフィック命令

## G PRINT.....グラフィックプリント

プログラム

## 省略形.....GP.

マニュアル



## 機能

指定されたドットパターンを表示します。

## 書式

- (1) GPRINT 文字列  
(2) GPRINT 式 [ ; 式 ; ... ]

## 参照

GCURSOR、PRINT

## 説明

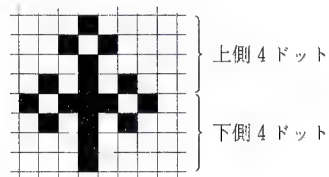
- 文字列や式で、表示させるドットパターンを指定します。ドットパターンは縦に並ぶ8ドットを1まとまりとして指定します。
- 書式(1)では、縦に並ぶ8ドットを下側4ドットと上側4ドットに分けて、それぞれのパターンを16進数で表し、文字列として「」で囲んで指定します。

|       | 16<br>進数 | ドット<br>パターン | 16<br>進数 | ドット<br>パターン | 16<br>進数 | ドット<br>パターン | 16<br>進数 | ドット<br>パターン |
|-------|----------|-------------|----------|-------------|----------|-------------|----------|-------------|
| 上側ドット | 0        |             | 4        |             | 8        |             | C        |             |
| 下側ドット | 1        |             | 5        |             | 9        |             | D        |             |
|       | 2        |             | 6        |             | A        |             | E        |             |
|       | 3        |             | 7        |             | B        |             | F        |             |

〈例〉 GPRINT "○○○○○○○○...."

数字2字を1組にして縦1列のドットパターンを指定します。  
最初の数字は下側4ドット、2番目の数字は上側4ドットを表します。

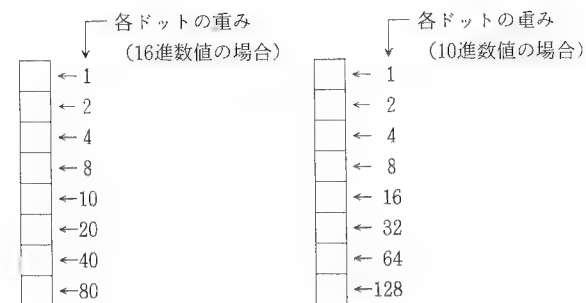
〈例1〉 GPRINT "102812FD122810"



0 8 2 D 2 8 0 ..... 上側のドットパターンを16進数で表した場合

1 2 1 F 1 2 1 ..... 下側のドットパターンを16進数で表した場合

- 書式(2)では、縦に並ぶ8ドットを1まとまりとして、ドットパターンを数値で指定します。縦に並ぶ8ドットは、下に示すようにそれぞれ“重み”が与えられています。



ドットパターンを指定するときは、点灯させたいドットの重みを加え合わせた値で指定します。

〈例2〉 例1に示したパターンを表示します。

16進数値による指定

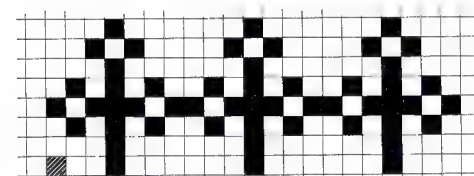
```
GPRINT &H10;&H28;&H12;&HFD;&H12;&H28;
&H10
```

10進数値による指定

```
GPRINT 16;40;18;253;18;40;16
```

- GCURSOR命令で表示開始位置が指定されている場合、GPRINT命令で表示される内容は、指定されている表示開始位置(ドット)を含んだ上側の8ドットを使って、最初のドットパターンを表示します。

〈例3〉 10 AAS="102812FD122810"  
20 GCURSOR(30,20)  
30 GPRINT AAS;AAS;AAS



↑  
GCURSOR命令の指定位置(30,20)

この位置より上側8ドットを使用して表示が開始されます。

- GPRINT命令がセミコロン(;)で終わっている場合は、実行後、表示された内容の1列右側が次の表示開始位置に指定されます。なお、GPRINT命令の最後がコロン、または改行の場合は、横方向(X方向)の指定が0に戻ります。
- WAIT命令で指定された時間だけ表示します。

## 基本命令

GRAD.....グラー度

プログラム

省略形.....GR.

マニュアル

## 機能

角度単位を“GRAD”(グラー度)に設定します。

## 書式

GRAD

- 参照** DEGREE、RADIANT
- 説明** ●三角関数、逆三角関数、座標変換で扱う角度の単位を“GRAD”単位[ $g$ ]に設定します。  
(1 直角=100 $g$ )

## 基本命令

HEX\$……ヘキサドル

プログラム

省略形……H.

マニュアル

- 機能** 数値データを10進数とみなし、16進数の文字列に変換します。
- 書式** HEX\$ 式
- 説明** 式の値は-999999999から999999999の範囲で、整数部のみが有効です。  
 〈例〉 C\$=HEX\$12+HEX\$15 C\$には"CF"が代入されます。

## 基本命令

IF~THEN~ELSE…イフ~ゼン~エルス

プログラム

省略形……IF T. EL.

- 機能** 条件を判断し、プログラムの流れ（実行の順番）を変えます。
- 書式**
- |             |                       |                                |
|-------------|-----------------------|--------------------------------|
| IF 条件式 THEN | { 行番号<br>ラベル<br>実行文 } | [ ELSE { 行番号<br>ラベル<br>実行文 } ] |
|-------------|-----------------------|--------------------------------|

- 参照** AND、OR
- 説明** ●IFに続く条件式が成立した場合はTHENに続く命令を実行します。条件式が成立しない場合は、ELSEに続く命令を実行します。ELSEを省略した場合、条件式が成立しないときは次の行へ実行が移ります。
- THENやELSEに続けて行番号またはラベルを書くと、その行またはラベルが書かれている行へジャンプします。
- 実行文として代入文を書くときは、THENまたはLET命令が必要です。
- THEN命令を省略して実行文（PRINT文、INPUT文など）を書くことができますが、ジャンプを指示する場合はTHENまたはGOTO命令が必要です。（ELSEは省略できません。）
- 具体的な使いかたについては、117ページも参照してください。
- 〈例〉 IF A<5 THEN C=A\*B:GOTO 50

もしAが5よりも小さければ、A\*Bの結果をCに代入して、50行へジャンプします。

IF B=C+1 GOTO 60 ELSE 100  
 または IF B=C+1 THEN 60 ELSE 100

もし、BとC+1の結果が等しければ60行へジャンプし、等しくなければ100行へジャンプします。

- 条件式としては、=、>、>=、<、<=、<>が使用できます。
- 条件式は、A+B+4のような変数や、通常の式にすることもできます。

IF B+4 THEN 60

IF A THEN 60

この場合は、変数や式の値が0以外（IF文が成立）ならばIF文の後に続く内容を実行し、0（IF文が不成立）ならば次の行を実行します。

- 条件式は次の例のように“\*”と“+”を用いて結合できます。

〈例〉 IF (A>5)\*(B>1) THEN……

Aが5よりも大きく、かつBが1よりも大きいとき、THENに続く内容を実行（論理積：AND）

IF (A>5)+(B>1) THEN……

Aが5よりも大きい、あるいはBが1よりも大きいとき、THENに続く内容を実行（論理和：OR）

- 文字列の比較

条件式に文字列を用いることにより、文字列の比較、大小判断ができます。

文字列はキャラクタコードの大きさによって比較されます。たとえば、キャラクタコードでは“A”が65、“B”が66、“C”が67……となっています。したがって“A”は“B”よりも小さく、“B”は“C”よりも小さくなります。同様に数字はアルファベットより小さくなります。

〈例〉 150行のデータを読み込んで、アイウエオ順に並べ替えます。

```
5  CLS
10  DIM Y$(4)
20  READ Y$(1), Y$(2), Y$(3), Y$(4)
30  GOSUB 200:PRINT
40  FOR A=1 TO 3
50  FOR B=A+1 TO 4
60  IF Y$(A)<=Y$(B) THEN 80
70  Y$(0)=Y$(A):Y$(A)=Y$(B):
   Y$(B)=Y$(0)
80  NEXT B:NEXT A
90  GOSUB 200:END
150  DATA アサミ, サチコ, アサコ, キヨミ
200  FOR A=1 TO 4
210  PRINT Y$(A);"___";
220  NEXT A
230  PRINT:RETURN
```

比較、並べ替えループ

- 1つの式で扱う文字列の長さについて  
 文字列の結合や大小比較などの式において、1つの式に含まれる文字（アルファベット、カタカナ、数字など）の数は、合計で255文字を超えない範囲で演算処理を行ってください。255文字を超えると、エラー55になります。

## 基本命令

IF~THEN~ELSE~ENDIF…イフ~ゼン~エルス~エンドイフ

プログラム

省略形……IF T. EL. ENDI.

- 機能** 条件を判断し、プログラムの流れ（実行の順番）を変えます。
- 書式** IF 条件式 THEN  
       実行文1  
       [ ELSE

実行文2]

ENDIF

実行文3

参照

説明

IF~THEN~ELSE、AND、OR、NOT、XOR

- IF、ELSE、ENDIFはそれぞれ行頭（ラインナンバーのすぐ後ろ）に書く必要があります。なお、ラベルの後ろは行頭にはなりません。
- THEN（もしくはELSE）の後ろにステートメント、式、リマーク等の実行文を書くことはできません。THENの後ろにステートメントなどを書いたり、THENを省略したりすると1行形式のIF~THEN~ELSEと見なされます。
- 条件式が成立した場合は、実行文1を実行した後、実行文3に移ります。条件式が成立しない場合は、実行文2を実行した後、実行文3に移ります。
- ELSEと実行文2を省略した場合、条件式が成立しないときは、実行文3に移ります。
- 条件式の使用法や判断内容はIF~THEN~ELSE（1行形式）の場合と同じです。

〈例〉 鶴亀算を行うプログラムです。

```

10 WAIT:CLS
20 INPUT "セノタイノカス" : I; A
30 LOCATE 14, 0: INPUT "アシノカス" : I; B
40 IF (4*A) < B OR (2*A) > B THEN
50   PRINT "IMPOSSIBLE ヤリナオシ!"
60 ELSE
70   C=B-INT(B/2)*2
80   IF C=1 THEN
90     PRINT "アシノカス" カ キスウテ ス
100  ELSE
110    X=(2*A)-B/2:Y=(B/2)-A
120    WAIT 0:PRINT "ツルノカス" ハ I; X
130    WAIT:PRINT "カメノカス" ハ I; Y
140  ENDIF
150 ENDIF
160 GOTO 10

```

①と②でブロック形式のIF~THEN~ELSE~ENDIFを使用しています。

## 関数

## INKEY\$…インキードル

プログラム

省略形…… INK.

機能

押されたキーの内容を読み込んで指定された文字変数に代入します。

書式

文字変数=INKEY\$

説明

- 通常、次の例のように繰り返しループを作って、有効なキーが押されるのを待ちます。実行されたときに、キーが押されていない場合は、変数にはNull（空白）が代入されます。

〈例〉

```

10 CLS
20 Z$=INKEY$
30 IF Z$="" THEN 20
40 PRINT "----"; ASC Z$; " "
50 Z$=INKEY$
60 IF Z$="" THEN 10 ELSE 50

```

このラインを繰り返し実行し、キーが押されるのを待ちます。

- <sup>BREAK</sup>ON はプログラムの一時停止キー（ブレークキー）として働きます。
- (SHIFT)を押しながら、キーを押したときに働く機能や入力される記号、(2nd F)に続いて押したときに働く機能などを読み込むことはできません。また、英小文字やカナ文字を読み込むこともできません。
- (OFF)、(SHIFT)は読み込むことはできません。
- 命令実行時に押されていたキーのデータを読み込むための命令です。INPUT命令のようにキー入力時に(ENTER)を押す必要はありません。
- (注) プログラムの初めにINKEY\$があると、プログラムをスタートさせたときにスタートキーを読み取ってしまうことがあります。

INKEY\$で読み取られるキーとキーコード

|    |                | 0     | 16    | 32    | 48 | 64 | 80               | ... | 128            | 144 | ... | 240 |
|----|----------------|-------|-------|-------|----|----|------------------|-----|----------------|-----|-----|-----|
|    | 16進上位<br>16進下位 | 0     | 1     | 2     | 3  | 4  | 5                | ... | 8              | 9   | ... | F   |
| 0  | 0              |       | 2nd F | SPACE | 0  |    | P                |     |                |     |     |     |
| 1  | 1              |       |       |       | 1  | A  | Q                |     |                | ln  |     |     |
| 2  | 2              | CLS   |       |       | 2  | B  | R                |     |                | log |     |     |
| 3  | 3              |       |       |       | 3  | C  | S                |     |                |     |     |     |
| 4  | 4              | ▲     | カナ    |       | 4  | D  | T                |     |                |     |     |     |
| 5  | 5              | ▼     | CAPS  |       | 5  | E  | U                |     |                | sin |     |     |
| 6  | 6              |       |       |       | 6  | F  | V                |     |                | cos |     |     |
| 7  | 7              | ANS   | BS    |       | 7  | G  | W                |     | 1/x            | tan |     |     |
| 8  | 8              | BASIC | R・CM  | (     | 8  | H  | X                |     | x <sup>2</sup> |     |     |     |
| 9  | 9              | TEXT  | M+    | )     | 9  | I  | Y                |     |                |     |     |     |
| 10 | A              | TAB   |       | *     |    | J  | Z                |     |                |     |     |     |
| 11 | B              | INS   |       | +     | ;  | K  |                  |     | →DEG           |     | π   |     |
| 12 | C              | CONST |       | ,     |    | L  |                  |     | F↔E            |     | √   |     |
| 13 | D              | ↩     |       | -     | =  | M  |                  |     | nPr            |     |     |     |
| 14 | E              | ▶     |       | .     |    | N  | y <sup>x</sup> △ |     | MDF            |     |     |     |
| 15 | F              | ◀     |       | /     |    | O  |                  |     |                |     |     |     |

## 基本命令

## INPUT……インプット

プログラム

省略形…… I.

機能

キーから数値または文字列の入力を行います。

書式


- (1) INPUT 変数 [, 変数……]
- (2) INPUT "文字", 変数 [, "文字", 変数……]
- (3) INPUT "文字"; 変数 [, "文字"; 変数……]

参照

INPUT#, INKEY\$, READ, LOCATE

説明

- 変数に、キーボードから数値や文字列を代入したいときに使用します。INPUT命令に続いて、データを格納するための変数を指定します。
- 書式(1)では、?を表示して入力待ち（プログラムの実行を停止）になります。データを入

力して、を押せば、データが変数に代入されて実行が再開されます。

(変数を複数個指定する場合はコンマ(,)で区切ります。)

- 書式(2)では、`' '`で囲まれた文字を入力ガイダンス(入力案内)として表示し、入力待ちになります。


データを入力すると入力ガイダンスは消えます。

「入力ガイダンスは、計算機がデータ待ちになったとき、何のデータを要求しているのか、どのデータを入力すればよいか、などをわかりやすくするためのメッセージです。」

- 書式(3)では、書式(2)の場合と同じく入力ガイダンスを表示して入力待ちになります。データを入力すると、入力ガイダンスに続けて、入力したデータが表示されます。

- 書式(1)、(2)、(3)は1つのINPUT命令の中で同時に使えます。

〈例〉 `INPUT 'A='; A, B, 'C=?', C`

- INPUT命令の入力待ちのときに、データを入力せずにのみを押したときは、変数に入っていたデータを保持したまま、次の実行に移ります。

- INPUT命令に続いて指定された変数の型と、入力するデータの型は同じでなければなりません。

文字変数に数値を入れると文字データとして認識されます。数値変数に文字を入れると、その変数に記憶されている数値が入力されます。

- INPUT命令実行前に、LOCATE命令により表示開始位置が指定されている場合は、その位置から入力ガイダンス、あるいは?が表示されます。

(注) INPUT命令による入力時のエラーは`CLS`で解除して、正しいデータを入れてください。

## ファイル関連命令

### INPUT #…インプット・クロスハッチ

プログラム

省略形……`I. #`

機能

ファイルのデータを、指定した変数に代入します。

書式

`INPUT #`ファイル番号, 変数[, 変数…]

参照

`OPEN`、`PRINT #`

説明

- SIO(シリアル入出力装置)に送られてくるデータ、またはラムデータファイル(データファイル)に記録されているデータを指定されている変数に代入します。
- この命令が有効になるのは、`OPEN`命令で“`COM:`”、“`COM1:`”を指定しているとき、または“`E:`”のINPUTを指定してオープンしているときだけです。
- `OPEN`命令で、“`COM:`”、“`COM1:`”を指定しているときは、ファイル番号を1に指定します。“`E:`”を指定しているときは、`OPEN`命令で指定したファイル番号(2または3)を指定します。
- `INPUT # 1, B(*)`とすると、配列全体の指定になります。文字配列変数のときは`CS(*)`のように指定します。くわしくは`PRINT #`命令を参照してください。
- 指定した変数の数よりもファイルのデータが少ない場合は、エラー87になります。

#### 変数への読み込み規則

- 数値

・データの区切りは、コンマ(,)およびスペース、`CR(&H0D)`、`LF(&H0A)`コードです。

・スペースでない最初の文字をデータの始まりとします。最初のスペースは無視します。

- ・数値化できないデータを読み込んだ場合は0とします。

- 文字

・データの区切りはコンマ(,)および`CR(&H0D)`、`LF(&H0A)`コードです。256文字目を読み込んだときも区切りになります。

・スペースでない最初の文字をデータの始まりとします。最初のスペースは無視します。

・データの始まりがダブルクォーテーション(“)のときは、次のダブルクォーテーションまでを1つのデータとします。

- EOF(エンドオブファイル)コードの処理(174ページで指定したコードです。)

・データの前にEOFコードがある場合はエラーになります。

・データの途中にEOFコードがある場合はデータの区切りと見なします。

## ファイル関連命令

### KILL……キル

省略形……`K.`

マニュアル

機能

プログラムファイルエリアに登録されているファイルを消去します。

書式

`KILL 'ファイル名'` 

説明

●プログラムファイルエリア内の指定したファイルを消去します。拡張子が`.`、`BAS`のときは拡張子の指定は省略できます。

●指定したファイルが存在しないときはエラー94になります。

## 基本命令

### LCOPY……ラインコピー

省略形……`LC.`

マニュアル

機能

プログラム行を複写します。(PROモードのマニュアル操作でのみ有効)

書式

`LCOPY` コピー元開始行番号, コピー元終了行番号, コピー先開始行番号

参照

`PASS`、`LIST`

説明

●コピー元開始行番号からコピー元終了行番号までのすべてのプログラム行を、コピー先開始行番号から複写します。

●`GOTO`、`GOSUB`、`RESTORE`命令などで参照している行番号は、コピー先のプログラムにおいても元のままです。必要に応じて変更してください。

●パスワードが設定されているときは、`LCOPY`命令は無視されます。

●次の場合はエラーになります。

- ・指定した行番号が存在しないとき
- ・コピー元開始行番号がコピー元終了行番号よりも大きいとき
- ・コピー先開始行がすでに存在するとき
- ・コピーしたプログラムがすでに存在するプログラムと行番号が混在するとき
- ・コピーした行番号が65279を超えると
- ・フリーエリアが足りなくなったとき

〈例〉 `LCOPY 10, 100, 200` 10行から100行までのプログラムを200行から同じ増分で複写します。



## 関数

## LEFT\$……レフトドル

プログラム

省略形……LEFT.

マニュアル

## 機能

文字列の左側から指定した文字数分を取り出します。

## 書式

LEFT\$ (文字列, 式)

## 参照

MID\$, RIGHT\$

## 説明

- 指定された文字列の左から、式の値で指定された桁数（文字数）だけ、文字を取り出します。たとえば、A\$ = "ABCDE" のとき LEFT\$ (A\$, 3) は A\$ の文字列の左側 3 文字、すなわち "ABC" を取り出しなさいという意味になります。
- 式の値は、0 ～ 255 の範囲の整数でなければなりません。

## 関数

## LEN……レングス

プログラム

省略形……なし

マニュアル

## 機能

文字列の文字数を求めます。

## 書式

LEN 文字列

## 説明

- 1 つの文字列の中に含まれる文字の数（記号、スペース、数字も含みます）を求めます。たとえば、A = LEN "ABC1234ハ" ン とすれば、文字数 10 が変数 A に代入されます。（だく点や半だく点も 1 文字と数えられます。）
- 文字列としては、AB\$ のように文字変数でも指定できます。

## 基本命令

## LET……レット

プログラム

省略形……LET.

## 機能

変数に数値や文字を代入するための命令です。

## 書式

[LET] 変数 = データ [ , 変数 = データ ] …

## 説明

- 代入文は LET 命令に続いて代入式を書きます。なお、LET 命令は省略できます。代入式は A = 5 + 3、B\$ = "ABC" のように、左辺に変数を、右辺に式や文字列を書きます。
- この場合の "=" は "等しい" という意味ではなく、"左辺の変数に、右辺の内容あるいは計算結果を入れなさい" という意味です。
- 変数とデータの型は同じ（文字型どうし、または数値型どうし）でなければいけません。

## ファイル関連命令

## FILES…エルファイルズ

マニュアル

省略形……LF.

## 機能

プログラムファイルエリアに登録されているファイルのファイル名を印字します。

## 書式

LFILLES 

## 参照

FILES

## 説明

- プログラムファイルエリアに登録されているすべてのファイル名をプリンタで印字します。

## グラフィック命令

## LINE……ライン

プログラム

省略形……LIN.

マニュアル

## 機能

指定された 2 点間を線で結びます。

## 書式

$$\text{LINE} [(式_1, 式_2)] [(式_3, 式_4)] [, \begin{Bmatrix} S \\ R \\ X \end{Bmatrix}] [, 式_5] [, \begin{Bmatrix} B \\ BF \end{Bmatrix}]$$

[A]                      [B]                      [C]                      [D]                      [E] 項

## 参照

GCursor、PSET

## 説明

- (式<sub>1</sub>, 式<sub>2</sub>) で指定される点と (式<sub>3</sub>, 式<sub>4</sub>) で指定される点を線で結びます。  
 <例> LINE (0, 0) - (143, 47)
- 式<sub>1</sub> ～ 式<sub>4</sub> の値の範囲は -32768 ～ 32767 ですが、画面に表示できる範囲は次のとおりです。  
 式<sub>1</sub>、式<sub>3</sub> : 0 ～ 143    画面の左上が (0, 0) で、右下が (143, 47) です。  
 式<sub>2</sub>、式<sub>4</sub> : 0 ～ 47  
 画面外の領域を指定した場合でも、-32768 ～ 32767 の範囲内であればエラーにならず、画面内に当たる部分のみが描かれます。
- (式<sub>1</sub>, 式<sub>2</sub>) を省略した場合は、(0, 0) 位置、または直前に実行された LINE 命令の (式<sub>3</sub>, 式<sub>4</sub>) で指定された位置から線が描かれます。  
 <例>    10 CLS  
       20 LINE (10, 0) - (143, 24)  
       30 WAIT: LINE - (71, 47)
- [C] 項の S、R、X により、線に当たるドットを点灯させるか、消すか、あるいは反転させるかを指定します。  
 S……線を描くとき、ドットを点灯させて線を描きます。（ドットをセット）  
 R……線を描くとき、ドットを消灯させて（消して）線を描きます。（ドットをリセット）  
       線の回りのドットが点灯している場合に線を描くときや、描かれている線を消すときなどに用います。  
 X……線を描くとき、線に当たるドットが点灯しているときは消して、消えているときは点灯させます。（ドットを反転）  
 指定を省略した場合は、S を指定したときと同じになります。
- [D] 項の式<sub>5</sub> の値により、線の種類を指定します。  
 たとえば、式<sub>5</sub> の値が 5503 (&H157F) の場合、次のような線が描かれます。



16ドット

左と同じ形が繰り返されて線が描かれます。

この 5503 (&amp;H157F) を 16 桁の 2 進数で表せば、次のようになります。

0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 1 (7F15 の順になります。)

上記の図で示した 16 ドット分と、この 2 進数を比べてみると、各桁の 1 に相当するドットが点灯し、0 に相当するドットが消えていることがわかります。

このように、式<sub>5</sub> の値を 16 桁の 2 進数に変換したときの各桁が 0 か 1 かによって、線の種類が指定されます。

したがって、式<sub>5</sub>の値が0のときは線が画面にあらわれず、65535 (&HFFFF) のときは実線になります。また、式<sub>5</sub>を省略した場合も実線になります。

ただし、〔C〕項でRが指定されているときは、各桁の1に相当するドットをリセット（消去）し、Xが指定されているときは、各桁の1に相当するドットの反転を行います。

- 式<sub>5</sub>の値は0～65535 (&HFFFF) の範囲で指定できます。
- 〔E〕項のB、BFにより、(式<sub>1</sub>、式<sub>2</sub>)の点と(式<sub>3</sub>、式<sub>4</sub>)の点を結ぶ線を対角線とした長方形を描きます。

B……長方形を描く

BF……長方形を線で塗りつぶすように描く

(注)

画面はドットで構成されていますので、斜めの線などは正確な直線にならない場合があります。また、曲線も正確な曲線にはなりません。

```

<例> 10 CLS :WAIT 0
      20 AAS='102812FD122810'
      30 GCURSOR(64,28)
      40 GPRINT AAS;AA$;AAS
      50 LINE(24,0)-(124,47),&HF18F,B
      60 LINE(34,3)-(114,44),X,BF
      70 GOTO 60
  
```

## 基本命令

### LIST……リスト

省略形……L.

マニュアル

機能

書式

記憶されているプログラムを表示させます。(PROモードのマニュアル操作でのみ有効)

- (1) LIST
- (2) LIST 行番号
- (3) LIST ラベル

参照

説明

LLIST、PASS

- 書式(1)では、プログラムの先頭行から、表示できる範囲で表示します。
- 書式(2)では、指定した行番号の行から、表示できる範囲で表示します。指定した番号の行がない場合は、それよりも大きく、かつ一番近い行から表示します。
- 書式(3)では、指定したラベルの書かれている行から、表示できる範囲で表示します。ラベルは163ページを参照ください。
- プログラムが記憶されていないときや、パスワードが設定されているときは、LIST命令は無視されます。
- プログラム内にないラベルや、プログラムの最終行よりも大きい行番号を指定した場合はエラー40になります。

## プリンタ命令

### LLIST……ラインリスト

省略形……LL.

マニュアル

機能

書式

プログラムをプリンタで印字します。(PROおよびRUNモードのマニュアル操作でのみ有効)

- (1) LLIST
- (2) LLIST { 行番号 }   
                  { ラベル }

参照

説明

(3) LLIST [開始行] - [終了行]

LIST、PASS

- 書式(1)では、計算機内のプログラムをすべてプリンタで印字します。
- 書式(2)では、指定した行番号または指定したラベルのついている行だけを印字します。
- 書式(3)では、指定した開始行から終了行までのプログラムを印字します。(開始行および終了行はラベルも可)  
なお、書式(3)では開始行または終了行の指定を省略できます。(同時に両方を省略することはできません)  
開始行を省略した場合は、プログラムの先頭行から、指定した終了行までのプログラムを印字します。  
終了行を省略した場合は、指定した開始行から最終行までのプログラムを印字します。  
〈例〉 LLIST -200 先頭行から200行までを印字します。  
          LLIST 100- 100行から最後の行までを印字します。
- 指定した番号の行がない場合は、それぞれの値よりも大きくかつ最も近い行が指定されます。ただし、開始行が終了行よりも大きくなるような指定をするとエラー44になります。
- パスワードが設定されているときは、LLIST命令は無視されます。

## ファイル関連命令

### LNINPUT #…ラインインプット・クロスハッチ

プログラム

省略形……LNI.#

機能

書式

参照

説明

ファイルの1行(255バイト以内)単位のデータを、指定した文字変数に代入します。

LNINPUT #ファイル番号, 文字変数[, 文字変数……]

OPEN

- SIO(シリアル入出力装置)に送られてくるデータ、またはラムデータファイル(データファイル)に記録されているデータを指定されている文字変数に代入します。
- この命令が有効になるのは、OPEN命令で“COM:”, “COM1:”を指定しているとき、または“E:”のINPUTを指定してオープンしているときだけです。
- OPEN命令で、“COM:”, “COM1:”を指定しているときは、ファイル番号を1に指定します。“E:”を指定しているときは、OPEN命令で指定したファイル番号(2または3)を指定します。

変数への読み込み規則

- データの区切りは、CR(&H0D)+LF(&H0A)コードです。  
256文字目を読み込んだときも区切りになります。SIOに送られてくるデータの区切りはシリアル入出力の条件設定に従います。  
CRとLFコードは、文字変数には代入されません。
- EOF(エンドオブファイル)コードの処理
  - ・データの前にEOFコードがあるときは、エラーになります。
  - ・データの途中でEOFコードがあるときは、EOFコードを読み込んだ時点で、通信が終了します。

## ファイル関連命令

### LOAD……ロード

省略形……LO.

マニュアル

機能

書式

参照

プログラムファイルエリアに登録されているBASICプログラムを呼び出します。

LOAD #ファイル名

SAVE

## 説明

- プログラムファイルエリアから、指定したファイル名のプログラムを呼び出します。
- 拡張子が「.BAS」のときのみ、拡張子の記述は省略できます。
- 読み込み終了時、SIOがオープンしているとクローズされます。
- テキスト(TEXT)プログラムを呼び出そうとするとエラー96になります。

## 基本命令

## LOCATE…ロケート

プログラム

省略形……LOC.

## 機能

表示の開始位置(ポジション)を指定します。

## 書式

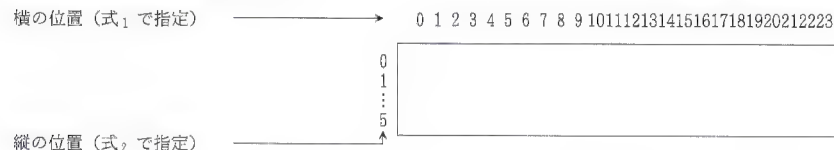
LOCATE 式<sub>1</sub> [, 式<sub>2</sub>]

## 参照

CLS、INPUT、PRINT

## 説明

- PRINT命令などで表示される内容の表示開始位置(カーソルの位置)を指定します。
- 表示位置は、次の図のようになります。

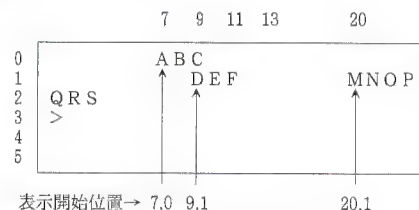


このように、表示部を横と縦に分け、式<sub>1</sub>の値で横の位置を指定し、式<sub>2</sub>の値で縦の位置を指定します。

- 式<sub>1</sub>の値は0～23、式<sub>2</sub>の値は0～5の範囲で指定します。この範囲外ではエラー33になります。
- 式<sub>2</sub>を省略した場合、縦の位置は現在カーソルがある位置になります。

〈例〉 5 CLS  
10 LOCATE 7, 0:PRINT "ABC"  
20 LOCATE 9, 1:PRINT "DEF"  
30 LOCATE 20, 1:PRINT "MNOPQRS"

プログラムを実行すると、次のように表示されます。



- LOCATE命令で表示開始位置を指定した場合は、表示の一部分だけを変えることもでき、応用範囲も広がります。スペースに変えると、一部分だけを消したことになります。
- LOCATE命令による指定はINPUT命令に対しても働きます。

## 基本命令

## LOF……エル・オー・エフ

プログラム

省略形……なし

マニュアル

## 機能

指定したラムデータファイルの未使用領域の大きさを求めます。

## 書式

LOF ファイル番号

## 説明

- ファイル番号で指定したラムデータファイルの未使用バイト数を求めます。
- 指定したファイル番号のファイルがOPENされていないとエラー85になります。

## プリンタ命令

## LPRINT…ラインプリント

プログラム

省略形……LP.

マニュアル

## 機能

指定した内容をプリンタで印字します。

## 書式

LPRINT [ { 式  
文字列 } [ { ; } { 式  
文字列 } ..... ] [ ; ]

## 参照

PRINT、USING

## 説明

- 項目を1つだけ指定したときは、式の値は紙の右側に詰めて印字し、文字は紙の左端から印字します。文字列が24桁を超える場合は自動的に改行して印字します。
- コンマ(,)を入れて2つ以上の項目を記述すると、1行の印字桁数24桁を左右12桁に分けて印字します。このときも12桁の範囲内で数値は右詰め、文字は左詰めにします。なお、印字内容が12桁を超える場合、数値は仮数部の下位桁を切り捨てて12桁以内で印字し、文字は先頭から12桁を印字します。
- 区切りセミコロン(;)を使用している場合は紙の左端から連続的に印字します。印字内容が24桁を超える場合は自動的に改行されます。
- プログラム中で末尾がセミコロン(;)の場合は、印字が終了しても改行を行わず、次のLPRINT命令で、指定されている内容を前の内容に続けて印字します。
- LPRINTのみで、印字する内容が指定されていないときは改行を行います。

## 関数

## MID\$……ミッドドル

プログラム

省略形……MI.

マニュアル

## 機能

文字列の中から指定した文字数分を取り出します。

## 書式

MID\$(文字列, 式<sub>1</sub>, 式<sub>2</sub>)式<sub>1</sub>: 文字列の左何文字目から取り出すかを指定します。式<sub>2</sub>: 何文字分を取り出すかを指定します。

## 参照

LEFT\$, RIGHT\$

## 説明

- 式<sub>1</sub>は1～255の範囲で指定できます。
- 式<sub>2</sub>は0～255の範囲で指定できます。ただし、0を指定した場合、文字は得られずNullになります。

〈例〉 10 AS="ABCDE"  
20 BS=MID\$(A\$, 2, 3) {A\$の文字列の左側2文字目から3文字、つまりBCDを取り出し、B\$に代入します。}  
30 PRINT B\$

**MON…………モニタ**

省略形……MO.

マニュアル

**機能** 機械語モニタモードにします。(RUNおよびPROモードのマニュアル操作でのみ有効)**書式** MON **説明** 機械語モニタモードにします。274ページ 「機械語モニタ機能」を参照ください。**基本命令****NEW…………ニュー**

省略形……なし

マニュアル

**機能** プログラムとデータを消去します。(PROモードのマニュアル操作でのみ有効)**書式** NEW **参照** CLEAR、PASS

**説明**

- プログラム・データエリア内のプログラム(BASICプログラム)や配列変数、単純変数がすべて消去され、固定変数の内容も消去されます。
- オープンしているファイル(デバイス)をクローズします。
- パスワードが設定されているときはNEW命令が無視されます。

**関数****NOT…………ノット**

省略形……NO.

プログラム

マニュアル

**機能** 与えられた数値の否定を取ります。**書式** NOT 式**参照** AND、OR**説明** ● 2進数において、否定は次の値を取ります。

NOT 1 = 0

NOT 0 = 1

● 10進数の否定を取った場合は、その10進数を2進数に変換して各桁の否定を取り、その結果を10進数に変換します。

このときの10進数をXとしたとき、Xとその否定(NOT X)の間には次の関係があります。

NOT X = -(X + 1)

この関係式から

NOT 0 = -1

NOT -1 = 0

NOT -2 = 1

となります。

**基本命令**

**ON~GOTO……オン~ゴートウ**  
**ON~GOSUB…オン~ゴースブ**  
 省略形……O. G.、O. GOS.

プログラム

**機能** 式の値により、指定された行を選択して実行を移します。**書式** ON 式 { GOTO } 行番号<sub>1</sub> [, 行番号<sub>2</sub>] [, 行番号<sub>3</sub>] ……**参照** GOTO、GOSUB

**説明**

- ONに続く“式”の値が1であれば“行番号<sub>1</sub>”、2であれば“行番号<sub>2</sub>”、3であれば“行番号<sub>3</sub>”というように、“式”の値により指定されている“行番号”が決定され、GOTOやGOSUBの各機能を実行します。
- “式”の値は整数部のみが有効になります。
- “式”の値が1より小さいときや指定されている“行番号”の個数より大きいときは、本命令の次の命令へ実行が移ります。
- “行番号”はラベルを指定することもできます。(ラベルについては163ページ参照)

〈例〉

```

10 CLS
20 LOCATE 5, 0
30 INPUT "ハ`ンコ`ウ(1-3)? ", N
40 LOCATE 10, 2
50 ON N GOTO 100, 200, 300
60 CLS:GOTO 20
70 END
100 PRINT "FIRST"
110 GOTO 20
200 PRINT "SECOND"
210 GOTO 20
300 PRINT "THIRD"
310 GOTO 20

```

**ファイル関連命令****OPEN…………オープン**

省略形……OP.

プログラム

マニュアル

**機能** SIO(シリアル入出力装置)に対するデータの入出力、ミニI/Oへの出力、ラムデータファイルに対するデータの入出力を可能にします。**書式** (1) OPEN "COM:" :

(2) OPEN "COM1:" :

(3) OPEN "PIO:" :

(4) OPEN "E:ファイル名" FOR モード AS #ファイル番号

(5) OPEN "LPRT:" :

**参照** CLOSE

**説明** ● 書式(1)、(2)、(3)では、SIOに対する入出力を可能にします。(SIO回路をオープンします。)ファイル名を書くことや、入出力の指定はできません。

書式(1)は半二重通信、書式(2)は全二重通信、書式(3)は8ビットの入出力指定(371ページ参照)です。

● 書式(1)、(2)の入出力条件はTEXTモードのSIOで設定します。



- 書式(4)では、ラムデータファイル(シーケンシャルデータ)へのデータの入出力を可能にします。なお、OPEN命令実行前に、ラムデータファイルモードでInit機能を使って、ファイルの確保と容量の指定を行ってなければなりません。(181ページ参照)  
ファイルが確保されていないとエラーになります。
- 書式(4)で指定するモードは次のとおりで、ファイル番号は2または3を指定します。  
INPUT……データの読み出しを行います。  
OUTPUT……データの書き込みを行います。すでにデータがある場合は、データの書き替えになります。  
APPEND……データの追加書き込みを行います。
- 書式(5)では、ミニI/Oに対する出力を可能にします。(367ページ参照)
- 書式(4)以外で複数の回路(ファイル)を同時にオープンしておくことはできません。(すべて自動的にファイル番号1を使用します)  
どれか1つがオープンしているときにOPEN命令を実行するとエラー86になります。  
ただし、これとは別に、書式(4)では同時に2つのファイルをオープンしておくことができます。

## 関数

OR……………オア

省略形……なし

プログラム

マニュアル

機能

式と式との論理和を計算します。また、条件式の結合を行います。

書式

式 OR 式

条件式 OR 条件式

参照

AND、NOT、IF

説明

- 2進法において、論理和は次のような値を取ります。  
1 OR 1 = 1      0 OR 1 = 1  
1 OR 0 = 1      0 OR 0 = 0
- 10進数の論理和を求めた場合は、10進数を2進数に変換したうえで、各桁の論理和を求め、その結果を10進数に戻します。  
たとえば、41と27の論理和は次のように計算されます。  
41 OR 27 = 59  
OR  $\begin{pmatrix} 101001 \cdots 41 \\ 011011 \cdots 27 \\ 111011 \cdots 59 \end{pmatrix}$       41と27をそれぞれ2進数に変換し、各桁のORを取ります。  
そして、その結果を10進数に変換すれば59になります。
- 2つ以上の条件のうち、いずれかを満足するような条件を1つの式で表します。  
〈例〉 A < 0 OR A > 6      Aは0よりも小さいか、あるいは6よりも大きい。  
IF A = 1 OR B = 1 OR C = 1 THEN…  
A、B、Cのいずれかが1のとき、THENに続く命令を実行します。

## グラフィック命令

PAINT……………ペイント

省略形……PAI.

プログラム

マニュアル

機能

式と式との論理和を計算します。また、条件式の結合を行います。

書式

式 OR 式

条件式 OR 条件式

参照

AND、NOT、IF

説明

- 2進法において、論理和は次のような値を取ります。  
1 OR 1 = 1      0 OR 1 = 1  
1 OR 0 = 1      0 OR 0 = 0
- 10進数の論理和を求めた場合は、10進数を2進数に変換したうえで、各桁の論理和を求め、その結果を10進数に戻します。  
たとえば、41と27の論理和は次のように計算されます。  
41 OR 27 = 59  
OR  $\begin{pmatrix} 101001 \cdots 41 \\ 011011 \cdots 27 \\ 111011 \cdots 59 \end{pmatrix}$       41と27をそれぞれ2進数に変換し、各桁のORを取ります。  
そして、その結果を10進数に変換すれば59になります。
- 2つ以上の条件のうち、いずれかを満足するような条件を1つの式で表します。  
〈例〉 A < 0 OR A > 6      Aは0よりも小さいか、あるいは6よりも大きい。  
IF A = 1 OR B = 1 OR C = 1 THEN…  
A、B、Cのいずれかが1のとき、THENに続く命令を実行します。

機能

指定した点を囲む領域を、指定した模様で塗りつぶします。

書式

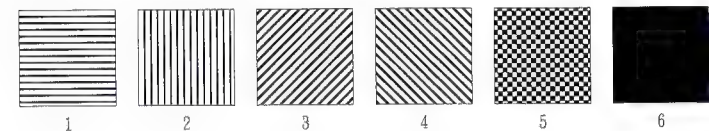
PAINT (式<sub>1</sub>, 式<sub>2</sub>), 式<sub>3</sub>

参照

CIRCLE、GCURSOR、LINE

説明

- (式<sub>1</sub>, 式<sub>2</sub>)で指定した点を囲む領域を、式<sub>3</sub>で指定した模様で塗りつぶします。  
なお、CIRCLE命令で円の内部を塗りつぶしたときの模様や表示している文字も境界になります。
- 式<sub>1</sub>、式<sub>2</sub>で指定できる範囲は-32768~32767です。画面内は、式<sub>1</sub>は0~143、式<sub>2</sub>は0~47の範囲です。
- 画面外の点を指定したときはPAINT命令は無視されます。
- PAINT命令を実行するためには、ワーク用として約1440バイト以上のフリーエリアが必要です。
- 式<sub>3</sub>で塗りつぶす模様を次のように指定します。



## 基本命令

PASS……………パス

省略形……PA.

マニュアル

機能

パスワードの設定あるいは解除を行います。(PROおよびRUNモードのマニュアル操作でのみ有効)

書式

PASS '文字'

参照

NEW、SAVE、BLOAD、BSAVE

説明

- 作成したプログラムを他の人に知られたくないときや変更されたくないときに、ある言葉や記号をパスワード(暗号)とし、パスワードを与えないかぎり、計算機内のプログラムを呼び出せないようにできます。(プログラムの秘密化)
- パスワードは、8文字までの英文字、カナ文字、数字、記号を使用することができます。'(Null)はパスワードとして設定できません。8文字以上のパスワードを宣言したときは、頭から8文字のみが有効となり、設定または解除が行われます。
- パスワードが宣言されていない状態のとき、PASS命令を実行すれば、そのとき計算機内にあるBASICプログラムに対してパスワードが設定され、秘密プログラムになります。
- 秘密化されたプログラムに対しては、LIST命令、やなど、プログラムの呼び出しにかかわる命令や機能、行の追加・削除などの機能は動きません。
- 秘密プログラムはNEW、DELETE命令でも消去されず、保護されます。AUTO、RENUM、LCOPY、LLIST命令は無視されます。また、プログラムファイルエリアへ登録したりポケコンや他の出力機器に出力することもできません。
- 秘密プログラムを解除する場合はもう一度同じパスワードを宣言します。(パスワードが違っているとエラー92になります。)
- 計算機内にBASICプログラムが入っていないときにPASS命令を実行するとエラー14になり、パスワードは設定されません。

## 関数

## PEEK……ピーク

プログラム

## 省略形……PE.

マニュアル

## 機能

機械語プログラムやデータを直接読み出します。

## 書式

PEEK 番地

## 参照

POKE、CALL

## 説明

- 指定した番地からデータを読み出します。
  - 番地は0～65535 (&H0～&HFFFF) の値で指定します。
  - 読み出されるデータは、0～255 (&H0～&HFF) の値になります。
  - パスワードが設定されているとき、本命令をマニュアルで実行するとエラー93になります。
- 〈例〉 4001番地 (16進表記) のデータを読み出し、変数Aに入れます。
- ```
A=PEEK &H4001
```

## グラフィック命令

## POINT……ポイント

プログラム

## 省略形……POI.

マニュアル

## 機能

指定したドットの状態を読みとる命令です。

## 書式

POINT (式<sub>1</sub>, 式<sub>2</sub>)

## 参照

GCURSOR、PSET、PRESET

## 説明

- (式<sub>1</sub>, 式<sub>2</sub>) で指定されたドットが点灯しているときは1、消えているときは0が得られます。指定されたドットが画面の範囲外にある場合は-1が得られます。
  - 式<sub>1</sub>、式<sub>2</sub>の値は-32768～32767の範囲内で指定できます。ただし、画面内は式<sub>1</sub>が0～143、式<sub>2</sub>が0～47の範囲になります。
- 〈例〉
- ```
10 CLS :WAIT 3:A=50
20 LINE (20, 8)-(20, 39) ← 画面に縦の線を2本描きます
30 LINE (100, 8)-(100, 39)
40 PSET (A, 24) ← 2本の線の間にドット (点) を点灯させます

50 B=POINT (A+1, 24) ← 右側のドットが点灯しているかどうかを調べます

60 IF B THEN 150 ← もし、点灯していたら150行へ行きます

70 PSET (A+1, 24) ← 消えているときは、そのドットを点灯させます

80 PRESET (A, 24) ← そして、前に点灯していたドットを消します

90 A=A+1 ← 横方向の位置を右へ寄せます
100 GOTO 50 ← 50行に戻ります
150 B=POINT (A-1, 24) ← 左が点灯しているかどうかを調べます

160 IF B THEN 50 ← もし、点灯していたら50行へ行きます
```

- ```
170 PSET (A-1, 24) ← 消えているときは、そのドットを点灯させます
180 PRESET (A, 24) ← そして、前に点灯していたドットを消します
190 A=A-1 ← 横方向の位置を左に寄せます
200 GOTO 150 ← 150行へ戻ります
```

このプログラムを実行すると、画面に描かれた2本の線の間にドットが行ったり来たりします。

## 基本命令

## POKE……ポーク

プログラム

## 省略形……POK.

マニュアル

## 機能

機械語プログラムやデータをメモリに直接書き込みます。

## 書式

POKE 番地, データ1, データ2, ……

## 参照

CALL、PEEK

## 説明

- 指定した番地をデータ記憶の開始番地として、データ1、データ2……と、順にメモリに記憶していきます。
  - 番地は0～65535 (&H0～&HFFFF) の値で指定します。
  - データは1バイト単位で指定します。したがって、各データの範囲は0～255 (&H00～&HFF) です。
  - パスワードが設定されているとき、本命令をマニュアルで実行するとエラー93になります。
- 〈例〉 &H01、&H02、&H03を7000～7002番地 (16進表記) に書き込みます。
- ```
POKE&H7000,&H01,&H02,&H03
```

## ご注意

この命令を誤って使用するとBASICプログラムやシステムエリアを破壊し、異常が発生することがあります。

## グラフィック命令

## PRESET……ポイント・リセット

プログラム

## 省略形……PRE.

マニュアル

## 機能

画面上の指定されたドット (点) を消します。

## 書式

PRESET (式<sub>1</sub>, 式<sub>2</sub>)

## 参照

PSET、GCURSOR、POINT

## 説明

- (式<sub>1</sub>, 式<sub>2</sub>) で指定されたドットを消します。
  - 式<sub>1</sub>、式<sub>2</sub>の値は-32768～32767の範囲内で指定できます。ただし、画面内は式<sub>1</sub>が0～143、式<sub>2</sub>が0～47の範囲になります。
- 〈例〉
- ```
10 CLS :WAIT 0
20 LINE (0, 0)-(143, 47), BF
30 FOR I=-1 TO 1 STEP 2
40 FOR X=-24 TO 24 STEP 0.5
50 Y=I*SQR ABS (24*24-X*X)
60 PRESET (X+71, Y+24)
```

```
70 NEXT X:NEXT I
80 WAIT :GPRINT
```

このプログラムを実行すると、塗りつぶされた四角形の中に円が描かれます。

## 基本命令

### PRINT……プリント

プログラム

### 省略形……P.

マニュアル

機能

指定した内容を表示部に表示します。

書式

```
PRINT [ { 式 } [ { , } { 式 } ..... ] [ ; ] ]
```

参照

LOCATE、LPRINT、USING、WAIT

説明

●項目を1つだけ指定したときは、式の値は表示部の右側に詰めて表示し、文字は表示部の左端から表示します。

〈例〉 10 PRINT 'ABCD'  
20 PRINT 123

```
RUN
ABCD
123.
```

ただし、LOCATE命令により、表示開始位置が指定されているときは、その位置から表示します。

●コンマ(,)で区切って2つ以上の項目を指定したときは、表示部を12桁ずつに区切り、最初に指定されている内容から順番に表示していきます。この場合も、12桁の範囲内で式の値は右側に詰めて表示し、文字は左側から表示します。なお、数値または文字が12桁を超える場合は次のように処理されます。

①数値が12桁を超える場合(指数表示において、仮数部が7桁以上になる場合)は、仮数部の下位桁が切り捨てられます。

②文字が12桁を超える場合は、先頭から12桁のみを表示します。

〈例〉 10 CLS  
20 A=123:B=5/9:C\$= 'ABCD'  
30 PRINT 'A=', A  
40 PRINT A,C\$, B

```
A= 123. 123.
5.555555E-01 123.ABCD
↑
仮数部の下位桁を切り捨て
```

●区切りにセミコロン(;)を使用している場合は、指定された内容を続けて表示します。

〈例〉 10 CLS  
20 A=123:B=5/9:C\$= 'ABCD'  
30 PRINT 'A='; A  
40 PRINT A;C\$;B;'VWXYZ'

```
A=123.
123.ABCD5.55555555556E-01V
WXYZ
```

●末尾がセミコロン(;)の場合は、その前に指定されている内容を左に詰めて表示し、その表示した内容の最後に続く桁が、次のPRINT命令に対する表示開始位置となります。

〈例〉 10 A=123:B=45  
20 CLS: PRINT  
30 C=A\*B

```
123*45=5535.
↑
この桁からCの内容を表示
```

### 40 PRINT C

- PRINTのみで、表示する内容が指定されていないときは、改行を行います。
  - LOCATE命令や、末尾がセミコロンのPRINT文で表示開始位置が指定されている場合は、その位置から表示を開始します。
- なお、このとき表示する内容の項目が(,)で区切られている場合、最初の項目は12桁の範囲にこだわらずに表示されます。
- 1つのPRINT命令で表示に使用する桁数は、255桁までです。255を超えた桁は切り捨てられます。

## ◎PRINT→LPRINT指定

本機はPRINT命令を、必要に応じて印字命令に切り替えることができます。

たとえば、計算機本体のみで使用しているときは、PRINT命令を表示命令として画面に計算結果を表示させ、プリンタを接続しているときは印字命令として計算結果などをプリントさせることができます。

## 指定・解除

プリンタが接続されているとき、マニュアルあるいはプログラムで

```
PRINT=LPRINT
```

の命令を実行すると、PRINT命令はすべてLPRINT命令と同様に働きます。

この機能は

```
PRINT=PRINT
```

の命令で解除できます。RUN命令の実行、**SHIFT**+**CLS**の操作、電源のオフ・オンなどでも解除され、PRINT命令は通常の表示命令に戻ります。マニュアルでPRINT=LPRINT命令を実行させ、有効に働かせるには、次の方法を用います。

- 命令実行後、GOTO命令でプログラムをスタートさせる。(163ページ参照)
- INPUT命令などにより、プログラムがストップしているときに、この命令を実行する。

## ファイル関連命令

### PRINT #…プリント・クロスハッチ

プログラム

### 省略形……P. #

マニュアル

機能

指定したデータをSIO(シリアル入出力装置)、ラムデータファイルに出力します。

書式

```
PRINT #ファイル番号, データ [ { ; } データ... ] [ { ; } ]
```

参照

OPEN、INPUT #

説明

- 指定したデータをSIOから出力します。またはラムデータファイル(シーケンシャルファイル)に記録します。
- この命令が有効になるのは、OPEN命令で“COM:”、“COM1:”を指定しているとき、または“E:”のOUTPUTあるいはAPPENDを指定してオープンしているときだけです。
- OPEN命令で、“COM:”、“COM1:”を指定しているときは、ファイル番号を1に指定します。“E:”を指定しているときは、OPEN命令で指定したファイル番号(2または3)を指定します。
- "データ"は、一般式(数式、文字式)の他に変数を指定できます。

PRINT # 1, B (\*) とすると、配列全体の指定になります。文字配列変数のときはC\$ (\*) のように指定します。

#### 配列全体の出力順序

〈例〉 一次元配列 B(0) → B(1) → B(2) → ……  
 二次元配列 C(0, 0) → C(0, 1) → C(0, 2) → ……  
 C(1, 0) → C(1, 1) → C(1, 2) → ……  
 :

- 配列全体を指定したときは、各要素を出力するごとにCR、LFコードを出力します。  
 たとえば、PRINT # 1, B (\*) はPRINT # 1, B (0) : PRINT # 1, B (1) : PRINT # 1, B (2) ……と指定した場合と同じ状態になります。
- 「データ」の後ろがコンマ (,) のときは20桁を1ゾーンとして出力します。  
 データが20桁以内なら1ゾーンの領域の中で、数値は右詰め、文字は左詰めで出力します。文字データが20桁を超えている場合は、必要なゾーン数を確保して出力します。ゾーンの残りの桁はスペースで埋めます。

〈例〉 A\$ = 'ABC' : C = 1 2 3 : D = -5. 2 E 1 0  
 PRINT # 1, A\$, C, D  
 出力 ABC\_\_\_\_\_

\_\_\_\_\_ 1 2 3. \_\_\_\_\_  
 -5. 2 E + 1 0 \_CRLF

↑ 出力フォーマットの説明参照  
 (下記)

- 「データ」の後ろがセミコロン (;) のときは、データを続けて出力します。  
 〈例〉 A\$ = 'ABC' : C = 1 2 3 : D = -5. 2 E 1 0  
 PRINT # 1, A\$; C; D  
 出力 ABC\_1 2 3. \_-5. 2 E + 1 0 \_CRLF
- 指定された「データ」をすべて出力すると、最後にCR (&H0D) とLF (&H0A) コードが送られます。ただし、データの最後にコンマ (,) またはセミコロン (;) が指定されているときはCR、LFコードは出力されません。
- 数値データの出力フォーマット
  - ① 数値データの後ろには1桁分のスペースがつけられます。
  - ② 数値の前には符号桁が1桁つきます。数値が負数の場合は、この桁が- (マイナス符号) になり、正数の場合はスペースになります。
  - ③ 指数形式で出力される場合は、仮数部の後ろに指数部を示す記号 (E)、符号、数値2桁 (1桁の場合は前に0をつける) が出力されます。  
 〈例〉 PRINT # 1, 1 2 3 4 5 6 7 8 9 8 7  
 \_1. 2 3 4 5 6 7 8 9 8 E + 1 0 \_CRLF を出力
  - ④ 未定義の変数 (CLEAR実行後のABなど) を指定した場合は0が出力されます。
- 文字データの出力フォーマット
  - ① 指定されているデータをそのままアスキー形式で出力します。
  - ② CHR\$(0) は " " (Null: 何もない状態) になります。
  - ③ 未定義の変数 (CLEAR実行後のAB\$など) を指定した場合は " " (Null) が出力されます。したがって、本機やパソコンでは何も表示されません。
- 文字や配列全体以外の文字変数を指定するとき、コンマ (,) やセミコロン (;) で続けるとデータの区切りコードがつかないため、ラムデータファイルから読み込むときにデータを分けることができません。したがって、次のように1命令に1データを指定するようにしてください。  
 PRINT # 1, 'ABC'

PRINT # 1, AB\$

## グラフィック命令

### PSET……ポイント・セット

プログラム

マニュアル

#### 省略形……PS.

機能

書式

参照

説明

画面上の指定されたドット (点) の点灯または反転を行います。

- (1) PSET (式<sub>1</sub>, 式<sub>2</sub>)
- (2) PSET (式<sub>1</sub>, 式<sub>2</sub>), X

PRESET、GCURSOR、POINT

- 書式 (1) では、(式<sub>1</sub>, 式<sub>2</sub>) で指定されたドットを点灯させます。
- 書式 (2) では、(式<sub>1</sub>, 式<sub>2</sub>) で指定されたドットが点灯しているときは消し、消えているときは点灯させます。
- 式<sub>1</sub>、式<sub>2</sub> の値は-32768~32767の範囲内で指定できます。ただし、画面内は式<sub>1</sub> が0~143、式<sub>2</sub> が0~47の範囲になります。

〈例〉 10 CLS :WAIT 0:DEGREE  
 20 FOR A=0 TO 420 STEP 3  
 30 B=-1\*SIN A  
 40 Y=INT(B\*24)+24  
 50 X=INT(A/6.25)  
 60 PSET(X,Y)  
 70 NEXT A  
 80 WAIT :GPRINT

このプログラムを実行すると、サインカーブが描かれます。

## 基本命令

### RADIAN…ラディアン

プログラム

マニュアル

#### 省略形……RAD.

機能

書式

参照

説明

角度単位を“RADIAN” (ラディアン) に設定します。

RADIAN  
 DEGREE、GRAD

- 三角関数、逆三角関数、座標変換で扱う角度の単位を“RADIAN”単位 [rad] に設定します。  
 (1直角=π/2 [rad])

## 基本命令

### RANDOMIZE…ランダムイズ

プログラム

マニュアル

#### 省略形……RA.

機能

書式

参照

RND命令の使用に先立って乱数のタネを植えつけるものです。

RANDOMIZE  
 RND



## 説明

- RND命令により乱数を発生させた場合、電源を入れ直すと常に同じ乱数系列を発生します。しかし、電源を入れた後にRANDOMIZE命令を実行すれば、実行のたびに違った乱数を発生させます。

## 基本命令

## READ……リード

プログラム

## 省略形……REA.

## 機能

DATA文に続いて指定されているデータを変数に読み込みます。

## 書式

READ 変数 [ , 変数] ……

## 参照

DATA、RESTORE

## 説明

- 変数にデータを代入する方法の1つです。変数はREAD文で指定し、データはDATA文で指定します。

〈例〉

```

5 0 READ A, B
1 1 0 DATA 3, 4, 5
1 2 0 DATA 6 0, 'G=', 'H='
2 0 0 READ C, D, E$, F$

```

Aには3、Bには4が代入されます。

Cには5、Dには60、E\$には'G='、F\$には'H='が代入されます。

- 1つのプログラムの中に何回でもREAD文およびDATA文を書くことができますが、データは何行に分けて書いても一連のデータと見なされ、小さい番号の行のデータから順番に変数に代入されます。また、複数のプログラムが書き込まれ、それぞれのプログラムにDATA文がある場合でも、それは一連のデータと見なされます。したがって、それぞれのプログラム内のデータを使用するときは、プログラムの最初の行に、RESTORE命令を書き込んでください。
- データと変数はその型（数値変数か文字変数かの型）が一致していなければなりません。
- 読み込もうとして、読み込むデータがない場合はエラー53になります。
- DATA文では文字データを' 'で囲まずに指定できますが、データの前後にスペースが指定されていても、スペースはないものと見なされます。

## 基本命令

## REM……リマーク

プログラム

## 省略形……なし

## 機能

プログラムに注釈を入れます。

## 書式

REM 注釈

## 説明

- プログラムの実行には関係なく、プログラムリストなどをわかりやすくする目的で、プログラムの先頭や途中に注釈を入れておくためのものです。
- 同じラインで、REMの後に書かれている内容はすべて注釈とみなされ、プログラム実行時は無視されます。
- REMの代わりに'（シングルクォーテーション）を使用することもできます。

〈例〉

```

1 0 REM キンリ ケイサン
:
```

2 0 0 サブルーチン

:

## 基本命令

## RENUM……リナンバー

マニュアル

## 省略形……REN.

## 機能

プログラムの行番号をつけ直します。（PROモードのマニュアル操作でのみ有効）

## 書式

RENUM [新行番号] [ , [旧行番号] [ , 増分]]

## 参照

PASS

## 説明

- “旧行番号”で指定したプログラムの行番号を“新行番号”に書き換え、それ以降の行番号を、“増分”で指定した値に従って順次書き換えていきます。
- “旧行番号”で指定した行番号がプログラム内に存在しない場合はエラー40になります。
- 新行番号、旧行番号、増分を省略した場合、旧行番号はプログラムの最初の行、新行番号、増分はそれぞれ10が指定されたものとみなされます。

〈例〉 RENUM

プログラム全体の行番号を10から10ステップきざみでつけ直します。

RENUM 1 0 0, 5 0, 1 0

行番号5 0が1 0 0になり、それ以後、行番号を10ステップきざみで最後までつけ直します。

- RENUM命令は、GOTO、GOSUB、IF~THEN~ELSE、ON~GOTO、ON~GOSUB、RESTORE命令などで参照している行番号も新しい行番号に対応させて自動的につけ直します。なお、参照する行番号がプログラム内に存在しないときはエラー40になります。
- 上記の命令で参照している行番号またはラベルの中に行番号やラベルでない文字や記号が入っていると、それより後に書かれている行番号のつけ直しは行いません。

〈例〉 GOTO 1 0 \* 2

ON A GOTO 'ABC', 2 0 0, AB, 4 0 0

～で示す部分は行番号やラベルとなりません。\_で示す部分は行番号とみなされ、つけ直しを行います。

- 新しくつけ直す行番号が65279を超える場合はエラー41になります。
- 実行の順番が変わるような指定（たとえば10、20、30の3つの行がある場合、RENUM15、30を実行するとき）を行うとエラー43になります。
- 長いプログラムに対してRENUM命令を実行すると、少し時間がかかります。表示部右端に\*を1つ表示しているときは<sup>BREAK</sup>ONで中断すると実行前のプログラムの状態に戻ります。\*を2つ表示しているときは<sup>BREAK</sup>ONは無視されます。
- パスワードが設定されているときは、RENUM命令は無視されます。

## 基本命令

## REPEAT~UNTIL…リピート~アンティル

プログラム

## 省略形……REP. UN.

## 機能

REPEATとUNTILの間に書かれた命令を指定された条件が満たされるまで、繰り返し実行します。

## 書式

REPEAT

実行文1

## UNTIL 条件式

実行文2

参照

FOR～NEXT、WHILE～WEND

説明

- REPEAT以降の命令（実行文1）を実行し、UNTIL文で条件式の判断をします。条件が成立した場合は、実行文2に移ります。（繰り返しループの終了）条件が成立しなかった場合は、実行文1を再実行し、条件が成立するまで繰り返します。（この繰り返し部分をREPEAT～UNTILループと呼びます。）
  - REPEATとUNTILは必ず対にして使用します。
  - REPEAT～UNTILループの中に、別のREPEAT～UNTILループを入れることができます。ただし、中に入るループは外のループ内に完全に入っていなければなりません。したがって、ループが交差するような使いかたはできません。（交差している場合はエラーになります。）この条件でループを最大22段まで重ねて使う（深みをもたせる）ことができます。（379ページのスタック欄を参照）
  - REPEAT～UNTILループの外からループ内に飛び込むことはできません。飛び込めるとエラーになります。
- （注）・REPEAT～UNTILループから外に飛び出した場合でも、そのループは終了したことにはなりません。プログラムによっては（REPEAT文を何回か実行するようなプログラムの場合）REPEAT～UNTILの深みエラーが発生することがあります。
- REPEAT～UNTILループ内ではCLEAR、ERASE、DIM命令は使用できません。

〈例〉 10 A=0 : B=0  
 20 REPEAT  
 30 A=A+B  
 40 INPUT B ← データ入力  
 50 UNTIL B<0 ← 負の値が入力されるまでREPEAT～  
 60 PRINT A UNTILを繰り返し実行します。  
 70 END

このプログラムはデータを入力しその累計を求めます。データとして負の値を入力すると、累計値を表示して終わります。

## 基本命令

## RESTORE…リストア

プログラム

## 省略形……RES.

機能

READ命令に続く変数に読み込まれるデータの順番を変えます。

書式

RESTORE [読み込み開始行]

参照

READ、DATA

説明

- READ命令の実行時、DATA命令で指定されているデータのどのデータを読み込むかは常に計算機に記憶されていますが、この読み込むデータの順番を強制的に変えるときに使用します。
- “読み込み開始行”を指定すると、その行のDATA命令または指定した開始行以降の最も小さい行番号のDATA命令から読み込みを開始します。
- “読み込み開始行”を省略するとプログラムの最初のDATA命令のデータから読み込みを開始します。
- “読み込み開始行”は、行番号またはラベルで指定します。

## 関数

## RIGHT\$…ライトドル

プログラム

## 省略形……RI.

マニュアル

機能

文字列の右側から指定した文字数分を取り出します。

書式

RIGHT\$ (文字列, 式)

参照

LEFT\$, MID\$

説明

- 指定された文字列の右から、式の値で指定された桁数（文字数）だけ、文字を取り出します。
- 式の値は0～255の範囲の整数でなければなりません。

〈例〉 10 AS="ABCDE"  
 20 BS=RIGHT\$(AS, 3) ← A\$の文字列の右側3文字すなわちCDEを取り出しB\$に代入します。  
 30 PRINT BS

## 関数

## RND……ランダム

プログラム

## 省略形……RN.

マニュアル

機能

乱数（疑似乱数）を発生させます。

書式

RND 式

参照

RANDOMIZE

説明

- RND XにおいてXの値により次のような乱数を得ることができます。

① Xが2以上の場合

Xが整数のとき：1からXの値以下の乱数を発生します。

(1 ≤ RND X ≤ X)

Xが小数部を含むとき：1からXの整数部に1を加えた値以下の乱数を発生します。

(1 ≤ RND X ≤ (INT X) + 1)

ただし、この場合は小数部の値によって乱数の発生に差が生じます。

② Xが負数の場合：同じ乱数（乱数列）を発生させるために、初期値を一定にします。

③ Xが0から2未満の場合：1未満の乱数を発生します。

(0 &lt; RND X &lt; 1)

- 乱数の有効桁数は10桁です。

〈例〉 10 CLS  
 20 USING "###"  
 30 FOR A=0 TO 2 40行は同じパターンの乱数を発生させるために  
 40 Y=RND -1 に入れています。  
 50 FOR B=0 TO 3 これにより、同じ乱数列が3回表示されます。  
 60 C=RND 9 40行を削除してプログラムを実行してみてください。  
 70 PRINT C; 同じ乱数が発生されなくなります。  
 80 NEXT B  
 90 PRINT: NEXT A  
 100 END

プログラムを実行すれば同じ乱数列が表示されます。


|   |   |   |   |             |
|---|---|---|---|-------------|
| 7 | 1 | 4 | 3 | } 3回表示されます。 |
| 7 | 1 | 4 | 3 |             |
| 7 | 1 | 4 | 3 |             |

## 基本命令

RUN…………ラン

省略形……R.

マニュアル

**機能** プログラムの実行を開始します。(RUNモードのマニュアル操作でのみ有効)**書式** (1) RUN (2) RUN { 行番号  
ラベル } **参照** GOTO

**説明**

- 書式(1)では、プログラムの一番小さい行番号から実行が開始されます。
- 書式(2)では、指定した行あるいはラベルが書かれている行からプログラムの実行が開始されます。(ラベルについては163ページ参照)
- 指定した行番号あるいはラベルがない場合はエラー40になります。
- プログラムに同じラベルが2個以上書かれているときは、行番号の小さいほうが実行されます。
- RUN命令によりプログラムの実行が開始されたときの計算機の状態については162ページをご覧ください。

## ファイル関連命令

SAVE…………セーブ

省略形……SA.

マニュアル

**機能** BASICプログラムをプログラムファイルエリアに登録します。**書式** SAVE 'ファイル名'**参照** LOAD、FILES

**説明**

- BASICプログラムにファイル名をつけて登録します。
- ファイル名は8文字以下の名前と、拡張子を3文字まで指定できます。
- 拡張子の記述を省略した場合は「.BAS」になります。
- 既存のファイル名を指定した場合は、そのファイルの書き換えになります。ただし、既存のファイル名がTEXT(テキスト)ファイルの場合はエラー96になります。
- 計算機内のプログラムが秘密化されているときや、プログラムがないときは、SAVE命令は無視されます。

## 基本命令

STOP…………ストップ

省略形……S.

プログラム

**機能** プログラムの実行を一時停止させます。**書式** STOP**参照** CONT

**説明**

- プログラムにバグ(誤り)が発生したとき、プログラムの実行を途中で止めて、変数の状態を調べたり、変数に数値などを代入したりしてバグを探します。このようなとき、プログラムを止めたい位置にSTOP命令を書き込みます。
- この命令が実行されると「BREAK IN 200」のように、実行した行番号を伝えるブレークメッセージを表示して停止します。
- この命令により停止したプログラムの実行を再開させるときはCONT命令を使用します。

## 関数

STR\$…………ストリングドル

省略形……STR.

プログラム

マニュアル

**機能** 数値を文字列に変換します。**書式** STR\$ 式**参照** VAL

**説明**

- 数値を文字列の形に変換します。  
たとえば、B=1234のとき  
A\$=STR\$ B  
として実行すれば、A\$には「1234」という文字列が代入されます。

## 基本命令

SWITCH~CASE~DEFAULT~ENDSWITCH  
…………スイッチ~ケース~デフォルト~エンドスイッチ  
省略形……SW. CAS. DEFA. ENDS.

プログラム

**機能** 変数の値に従って、処理の1つを実行します。**書式** SWITCH 変数CASE { 式  
文字列 }

実行文

[CASE { 式  
文字列 }

実行文]

[DEFAULT

実行文]

ENDSWITCH

**参照** ON~GOTO

**説明**

- 変数の値をCASEの式(文字列)と比較し、一致した場合にそのCASE以降の処理を実行します。1つの処理の範囲は次のCASE、DEFAULT、もしくはENDSWITCHまでです。1つの処理を実行すると、ENDSWITCHステートメントに移ります。また、変数の値がどのCASEとも一致しなかったときは、DEFAULTに移ります。DEFAULTを省略している場合はENDSWITCHに移ります。
- SWITCHとENDSWITCHは必ず対にして使用します。
- CASE、DEFAULT、ENDSWITCHは行頭(ラインナンバーのすぐ後ろ)に書く必要があります。なお、ラベルの後ろは行頭にはなりません。行頭に無い場合はSWITCHで認識されないだけでなく、実行した場合エラーになります。
- CASEに同じ値を指定してもエラーにはなりませんが、実行されるのはSWITCHに近いほうのCASE処理だけです。
- DEFAULTはCASEの処理をすべて書き終わってから一番最後に書かなければなりません。(DEFAULT処理の後にCASEを書くことはできません。)
- CASEまたはDEFAULTの実行文中に、SWITCH文は使用できません。
- SWITCH~ENDSWITCHの外から内部に飛び込むことはできません。飛び込ませるとエラーになります。

- (注) ・SWITCH～ENDSWITCHから外に飛び出した場合でも、SWITCH処理が終了したことはありません。プログラムによっては次のSWITCHでエラーが発生します。
- ・SWITCH文では深みをもたせることはできません。なお、スタックは6バイト以上残っていないと使用できません。(379ページのスタック欄参照)
- ・変数には文字変数や数値変数を使うことができますが、式は使用できません。

```

<例>  10 INPUT "ミセノナマエ"; A$
      20 SWITCH AS
      30 CASE "ABC"
      40 PRINT A$; "TEL: 012-3456"
      50 CASE "XYZ"
      60 PRINT A$; "TEL: 024-6802"
      70 DEFAULT
      80 PRINT A$; "ハ ミトウロク"
      90 ENDSWITCH
     100 END

```

このプログラムは、店の電話番号のクイックリファレンスです。中のデータと入力された店名を照合して、合えば店名と電話番号を表示し、合わなければ「ハ ミトウロク」と表示して終わります。

## 基本命令

### TROFF……トレースオフ

#### 省略形……TROF.

プログラム

マニュアル

**機能** トレースモードを解除します。

**書式** TROFF

**参照** TRON

**説明** ●TRON命令で設定されたトレースモードを解除します。(TRON参照)

## 基本命令

### TRON……トレースオン

#### 省略形……TR.

プログラム

マニュアル

**機能** トレースモードを設定します。

**書式** TRON

**参照** TROFF

- 説明** ●トレースモードでは、プログラムが1行実行されると、その実行された行番号を表示して停止します。
- 次の行を実行するときは[▼]を押します。[▼]でプログラムを1行ずつ実行させていくことができますので、プログラムがどのように実行されていくかをたどることができます。(156ページのデバック参照)
- これにより、プログラムが正しく実行されるかどうかをチェックできます。
- トレースモードはTROFF命令の実行、[SHIFT] + [CA]の操作、電源オフ・オンなどで解除されます。

## 基本命令

### USING……ユージング

プログラム

マニュアル

#### 省略形……U.

**機能** 数値や文字などを表示または印字するときのフォーマットを指定します。

**書式** (1) USING "フォーマット"

(2) USING

**参照** PRINT、LPRINT

- 説明** ●PRINT命令による表示の形、LPRINT命令による印字の形を指定する命令です。また、マニュアル計算の結果が数値の場合には、このUSINGのフォーマットに従います。
- 表示フォーマットは、書式(1)ではUSING命令に続く「」内に、次の記号を用いて指定します。指定の解除は書式(2)で行います。

|   |                                                                                                                                                                                                       |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| # | 数値の桁数を指定<br>整数部桁数: 2～11桁(符号を含む)<br>指定桁より数値のほうが少ないときはその分だけスペース表示になり、多いときはエラーになります。12桁以上指定した場合はすべて11桁の指定とみなされます。<br>小数部桁数: 0～12桁(指数方式による表示のときは0～9桁)<br>指定桁より数値のほうが少ないときはその分だけ0表示になり、多いときはその分だけ切り捨てられます。 |
| . | 小数点の表示を指定(整数部と小数部の区切りを指定)                                                                                                                                                                             |
| , | 数値の3桁区切りの指定<br>数値の整数部を3桁ごとにコンマ(,)で区切って表示するときは、整数部の#の途中または最後にコンマを記述します。                                                                                                                                |
| / | 数値の指数方式による表示を指定<br>整数部の指定桁にかかわらず、仮数部の整数桁は2桁(符号を含む)になります。                                                                                                                                              |
| & | 文字列の桁数を指定<br>指定桁より文字数が少ない場合はその分スペース表示になり、文字数が多い場合は指定された桁数分だけ表示します。                                                                                                                                    |

- <例>** ① USING "###" 符号と整数2桁を表示  
 ② USING "###." 符号と整数2桁と小数点を表示  
 ③ USING "###.###" 符号と整数2桁と小数点と小数点以下2桁を表示  
 ④ USING "###,###" 符号、3桁区切りマーク(,)、整数5桁と小数点を表示  
 3桁区切りマークも1桁と数えます。たとえば、  
 -1,234,567. を表示させるときはUSING "###,###.###" のように#と, をあわせて最低10個指定する必要があります。  
 ⑤ USING "##.##^" 小数点以下2桁までの指数方式で表示  
 [このとき、仮数部の整数は符号と整数1桁、指数部は符号を含めて4桁(E 00)が自動的に取られます。]  
 ⑥ USING "&&&&&" 文字を6桁表示  
 ⑦ USING "###&&&" 数値と文字を同時に指定  
 ⑧ USING フォーマット指定を解除

(注) コンマ(,)と^は混在しては使用できません。



- USING命令はPRINT文の中でも使用することができます。

〈例〉 10 B = -10, 8 : C = 10, 7703  
 20 PRINT USING '&&&###'; B = ; B,  
 'C = ' ; USING '###.###'; C

- USING命令で表示フォーマットの指定を行うと、以降に実行されるPRINT命令、LP  
 RINT命令には、すべてそのフォーマット指定が有効になります。

したがって、フォーマット指定が不要なときは書式(2)の形で指定を解除してください。  
 フォーマット指定はRUN命令の実行や、(SHIFT) + (CA) の操作などでも解除されます。

## 関数

## VAL…………バリュー

プログラム

## 省略形……V.

マニュアル

## 機能

文字列を数値に変換します。

## 書式

VAL 文字列

## 参照

STR\$

## 説明

- 数字(0~9)、符号(+、-)、指数部を示す記号(E)で構成されている文字列、および先頭に&Hがついた16進数を表す文字列を数値(10進数)に変換します。
- 1つの文字列の中に数値に変換できない文字や記号が含まれている場合、それから右の文字列は無視されます。

〈例〉 A = VAL ' - 120 '      Aに-120が代入されます。  
 B = VAL ' 3. 2 \* 4 = '      Bに3.2が代入されます。  
 C = VAL ' &HFF '      Cに255が代入されます。

## 関数

## VDEG…………バリュー・ディグリー

プログラム

## 省略形……VD.

マニュアル

## 機能

60進数(度・分・秒)の文字列を10進数(度)に変換します。

## 書式

VDEG 文字列

## 参照

DMS\$

## 説明

- 60進数の文字列を10進数に変換します。

〈例〉 10 AAS\$ = ' 1° 30' 36" '      RUN  
 20 B = VDEG AAS      > 1. 51  
 30 PRINT B

- 文字列に度(°)、分(')、秒(")の記号が含まれていないときは、整数部を度(°)、小数点以下1~2桁目を分(')、小数点以下3桁目以降は秒(")とみなします。

〈例〉 10 B = VDEG ' 1. 3036 '      RUN  
 20 PRINT B      > 1. 51

- 文字列の中に数値に変換できない文字や記号が含まれている場合は、エラー22が発生します。

- “VDEG DMS\$ A”と“DEG DMS A”の結果が異なる場合があります。  
 これは、DMS\$とDMSの有効桁数が違うためです。

DMS\$の有効桁数は10桁、DMSの有効桁数は12桁です。

〈例〉 10 A = 1. 23456789  
 20 PRINT DEG  
 DMS A  
 30 PRINT VDEG  
 DMS\$ A

```
RUN
1. 23456789
1. 234566667
>
```

- 度・分・秒の記号のキャラクターコード(くわしくは384ページ参照)

度(°) …223 (H&DF)、分(') …39 (H&27)、秒(") …248 (H&F8)

## 基本命令

## WAIT…………ウエイト

プログラム

## 省略形……W.

マニュアル

## 機能

PRINT命令によるプログラムの停止時間を指定します。

## 書式

(1) WAIT 式  
 (2) WAIT

## 参照

PRINT

## 説明

- 書式(1)では、式で時間を指定すると、PRINT命令を実行するたびに指定時間だけプログラムが停止し、その時間が経過すると自動的に再開します。
  - 式の値は0~65535の範囲で指定できます。  
 なお、式の値1は約1/64秒に相当します。
  - 時間を無限にしたいときは、書式(2)の形で時間指定を解除してください。この場合、プログラムを再開するには(↵)を押す必要があります。
  - 本機の電源を入れたとき、またはRUN命令を実行したときはWAIT 0(停止時間0)に設定されます。
  - (注) 一般のパソコンではWAIT指定はできません。パソコンでは通常次のようにFOR~NEXT EXTを用いて、時間の調整を行います。
- 〈例〉 50 FOR J = 1 TO 500 : NEXT J

## 基本命令

## WHILE~WEND…ホワイル~ホワイルエンド

プログラム

## 省略形……WH. WE.

## 機能

WHILEとWENDの間に書かれた命令を指定された条件が満たされている間、繰り返し実行します。

## 書式

WHILE 条件式  
 実行文1  
 WEND  
 実行文2

## 参照

FOR~NEXT、REPEAT~UNTIL

## 説明

- WHILE文において条件式の判断をし、条件が成立しなかった場合は実行文2に移ります。  
 (繰り返しループの終了)。条件が成立している間は実行文1を繰り返し実行します。(この繰

り返し部分をWHILE～WENDループと呼びます。)

- WHILEとWENDは必ず対にして使用します。
- WHILEの条件式が最初から成立していない場合は、1回も実行せずにループを抜けます。つまり、実行文1を1回も実行せずに実行文2に移ります。条件式がずっと成立している場合は永遠にループ内の実行を繰り返すプログラムになります。
- WHILE～WENDループの中に、別のWHILE～WENDループを入れることができます。ただし、中に入るループは外のループ内に完全に入っていないければなりません。したがって、ループが交差するような使いかたはできません。(交差している場合はエラーになります。)この条件でループを最大18段まで重ねて使う(深みをもたせる)ことができます。(379ページのスタック参照)
- WHILE～WENDループの外からループ内に飛び込むことはできません。飛び込ませるとエラーになります。

(注)・WHILE～WENDループから外に飛び出した場合でも、そのループは終了したことにはなりません。プログラムによっては(WHILE文を何回か実行するようなプログラムの場合)WHILE～WENDの深みエラーが発生することがあります。

- WHILE～WENDループ内ではCLEAR、ERASE、DIM命令は使用できません。

```
<例> 10 PRINT '3 カラ 100 マテ`ノ ソスウヲ モトメマス'
      20 DIM A(30):J=0:A=3:A(0)=2
      30 WHILE A<=100
      40   FOR I=0 TO J
      50     IF A-INT(A/A(I))*A(I)=0 THEN
      60       I=J:NEXT I:GOTO 90
      70   NEXT I
      80   PRINT A
      90   J=J+1:A(J)=A
      100  A=A+1
      110 WEND
      120 END
```

このプログラムは、3以上の数を2もしくは算出した素数で割り切れるかどうかで素数の判断をしています。A<=100(条件式が成立)の場合は素数を求める計算式を実行し、Aが100を超えるまでWHILE～WEND間を繰り返し実行します。

## 基本命令

### XOR……エクスクルーシブオア

省略形……X.

プログラム

マニュアル

機能

式と式との排他的論理和を計算します。

書式

式 XOR 式

条件式 XOR 条件式

参照

AND、OR、NOT、IF

説明

- 2進数において、排他的論理和は次のような値を取ります。

```
1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
```

0 XOR 0 = 0

なお、10進数の排他的論理和を求めた場合でも、計算機内では10進数を2進数に変換したうえで、各桁の排他的論理和を求め、その結果を10進数に戻します。たとえば、41と27の排他的論理和は次のように計算されます。

```
41 XOR 27 = 50
  101001.....41
XOR 011011.....27
-----
  110010.....50
```

41と27をそれぞれ2進数に変換し、各桁のXORを取ります。そして、その結果を10進数に変換すれば50になります。

- 式の値は-32768～32767の整数部が有効になります。
- 2つ以上の条件のうち奇数個を満足するような条件を1つの式で表します。

<例> 50 IF A>5 XOR B>=4 THEN...

A>5またはB>=4のいずれか一方のみが満足するとき、THENに続く命令を実行します。

本機には、制御実習のために、周辺機器接続端子(11ピン)を制御できるミニI/O、および8ビット制御に関する命令システムバス端子(40ピン)に対する入出力命令もあります。

これらの命令は本機で制御実習を行うとき以外には使用しないでください。詳しくは、先生の指導に従ってください。

## ミニI/Oに関する命令

### INP……イン・ポート

プログラム

マニュアル

省略形……なし

機能

入力ポート(Xin, Din, ACK)からの入力関数です。

書式

INP

参照

OUT

説明

- ポート制御命令で、INP命令は現在の入力ポートの状態を読み取って、0～7の値で返します。入力ポートはXin, Din, ACKの3つからなり、Xin=4、Din=2、ACK=1の重みを持ちます。すなわち、各入力ポートの状態を2進数の各桁と見なしたときの値を、10進数に変換して0～7の値を返します。なお、信号レベルはHi=1、Lo=0とします。

<例1> Xin=Lo, Din=Hi, ACK=Hiのとき

$0 \times 4 + 1 \times 2 + 1 \times 1 = 3$  で

INPの値は3となります。

<例2> 50 IF INP AND 2 THEN 50

Dinの値がLoレベルであれば次の行に実行が移り、Hiレベルであれば50行を繰り返し実行します。

(注)・OPEN'COM1:'またはOPEN'PIO:'が実行されているとエラーになります。

## ミニI/Oに関する命令

## OUT.....アウト・ポート

プログラム

省略形.....なし

マニュアル

**機能** 出力ポート (Busy, Dout, Xout) への出力命令です。**書式** OUT 式**参照** INP**説明** ●ポート制御命令で、OUT命令は10進数 (0~7) で指定した値を出力ポート Busy、Dout、Xout に出力します。指定する値は、出力ポートをそれぞれ Busy=4、Dout=2、Xout=1 の重みで加算した値です。すなわち、各出力ポートの状態を2進数の各桁と見なしたときの値を10進数で指定します。なお、信号レベルは Hi=1、Lo=0 とします。

〈例〉 OUT 6

 $6 = 1 \times 4 + 1 \times 2 + 0 \times 1$  で

Busy=Hi, Dout=Hi, Xout=Lo となります。

(注) ●OPEN #COM1: ' または OPEN #PIO: ' が実行されているとエラーになります。


●OUT命令で出力した内容は、次のOUT命令が実行されるまで保持されます。ただし、次の操作を行った場合は出力ポートの状態が変化します。

(1) 次の命令を実行したとき

|                    |         |       |
|--------------------|---------|-------|
| LPRINT             | PRINT # | END   |
| LLIST              | INPUT # | CLOSE |
| (SHIFT) + (P<=>NP) | RUN     | BSAVE |
| BLOAD              |         |       |

(2) その他、次の操作を行ったとき

- ・電源を入れた直後
- ・RUN実行直後
- ・マニュアル計算実行

ただし、あらかじめ OPEN #LPRT: '  と操作しておけば出力ポートの状態は保持されます。

- ・モードを切り替えた場合

## ミニI/Oに関する命令

## OPEN.....オープン

プログラム

省略形.....OP.

マニュアル

**機能** 出力デバイスを指定します。**書式** OPEN #LPRT: '**参照** CLOSE、LPRINT、LLIST**説明** ●OPEN #LPRT: ' 実行後のLPRINT、LLIST命令は、パラレルポートに対しての出力命令になります。

## ミニI/Oに関する命令

## CLOSE.....クローズ

プログラム

省略形.....CLOS.

マニュアル

**機能** デバイス指定を解除します。**書式** CLOSE**参照** OPEN、LPRINT、LLIST**説明** ●OPEN状態を解除し、それ以後のLPRINT、LLIST命令はCE-126P (プリンタ) に対しての命令になります。

## ミニI/Oに関する命令

## LLIST.....ラインリスト

マニュアル

省略形.....LL.

**機能** プログラム内容をパラレルポートより送出します。**書式** (1) LLIST(2) LLIST { 行番号 }  
                  { ラベル }

(3) LLIST 開始行~終了行

**参照** OPEN、CLOSE**説明** ●データ出力命令です。OPEN #LPRT: ' により、パラレルポートが指定されているときに、プログラムをアスキーコードで送出します。

- CLOSEされているときは、プリンタ (CE-126P) でプログラムを印字する命令になります。(341ページ参照)

## ミニI/Oに関する命令

## LPRINT...ラインプリント

プログラム

省略形.....LP.

マニュアル

**機能** 指定した内容をパラレルポートから送出します。**書式** LPRINT [ { 式 } [ { 文字列 } ] [ { ; } ] [ { 式 } ..... ] [ ; ] ]**参照** OPEN、CLOSE**説明** ●OPEN #LPRT: ' によりパラレルポートが指定されているときに、指定した内容をパラレルポートからアスキーコードで送出します。

- CLOSE (クローズ) されているときは、プリンタ (CE-126P) の印字命令になります。

## 8ビット制御に関する命令

## PIOSET…パラレルセット

プログラム

マニュアル

## 省略形……PI.

機能

書式

参照

説明

ミニI/Oで8ビット制御を行うために各信号の入出力モードを設定します。

PIOSET 式

PIOGET、PIOPUT

- 各信号を入力モードにするか、出力モードにするかの設定を行います。
- 式は8ビットに変換され、対応するビットが“1”のときは入力モードに設定され、“0”のときは出力モードに設定されます。

ビット7 EX2

ビット6 EX1

ビット5 ACK

ビット4 Din 〈例〉PIOSET &HF0とすると

ビット3 Xout Din、ACK、EX1、EX2の端子が入力モードに指定されます。

ビット2 Xin

ビット1 Dout 他は出力モードになります。

ビット0 Busy

- 式は0～255までの値で、この範囲外の値を指定するとエラーになります。0のときは全ての端子が出力モードになり、255のときは入力モードになります。

## 8ビット制御に関する命令

## PIOGET…パラレルインプット

プログラム

マニュアル

## 省略形……PIOG.

機能

書式

参照

説明

入力ポートからの8ビット入力関数です。

PIOGET

PIOSET、PIOPUT

- OPEN「PIO:」により8ビット制御が指定されているときに、PIOSET命令で入力モードに設定された端子からのデータを読み取ります。

値は、0～255になります。

- PIOSET命令で出力モードに設定されている端子は、常に“0”になります。

## 8ビット制御に関する命令

## PIOPUT…パラレルアウトプット

プログラム

マニュアル

## 省略形……PIOP.

機能

書式

参照

説明

出力ポートへの出力命令です。

PIOPUT 式

PIOSET、PIOGET

- OPEN「PIO:」により8ビット制御が指定されているときに、PIOSET命令で出力モードに設定された端子からデータを出力します。

1回の命令実行で0～255の範囲内の値を1回だけ返します。

- PIOSETで指定した出力端子のうち、式で指定したビット分だけマスクして出力します。

〈例〉PIOSETで15として指定しているとき、PIOPUTの式が15(00001111)のときは“0”しか出力されず、92(01011100)のときは80(01010000)が出力されます。

- PIOSET命令で入力モードに設定されている端子は、無視されます。

(注)・8ビット制御を使用する場合は、周辺機器接続端子(11ピン)がショートしないようにしてください。(1kΩ以下でショートすると故障の原因となります。)

## システムバス入出力命令

## INP……イン・ポート

プログラム

マニュアル

## 省略形……なし

機能

書式

説明

指定したポートから送られてくるデータを読み込みます。

INP ポートアドレス

- “ポートアドレス”で指定したポートから送られてきたデータを読み込みます。

- “ポートアドレス”は0～65535または&H0～&HFFFFの範囲で指定します。変数や式での指定はできません。

〈例〉A=INP &H20

ポート20Hから送られてきたデータを読み込み、変数Aに代入します。

- “ポートアドレス”を指定しないときはミニI/Oに対する命令になります。(366ページ参照)

## システムバス入出力命令

## OUT……アウト・ポート

プログラム

マニュアル

## 省略形……なし

機能

書式

説明

指定したポートへデータを送出します。

OUT ポートアドレス, データ

- “ポートアドレス”で指定したポートへ、“データ”で指定した値を出力します。

- “ポートアドレス”は0～65535または&H0～&HFFFFの範囲で指定します。

- “データ”は0～255の範囲で指定します。

〈例〉OUT 32, 121 ポート32(20H)へ121(79H)を出力します。

- “ポートアドレス”を指定しないときはミニI/Oに対する命令になります。(367ページ参照)

(注)・“ポートアドレス”で指定した値(アドレス)は、アドレスバス(A<sub>0</sub>～A<sub>15</sub>端子)へ出力されます。

“データ”で指定した値はデータバス(D<sub>0</sub>～D<sub>7</sub>端子)へ出力されます。

- “ポートアドレス”は20H～3FHを使用してください。他のアドレスはシステム(本体)で使用しています。



# 付 録

## 1. パソコン通信ケーブルCE-T800について

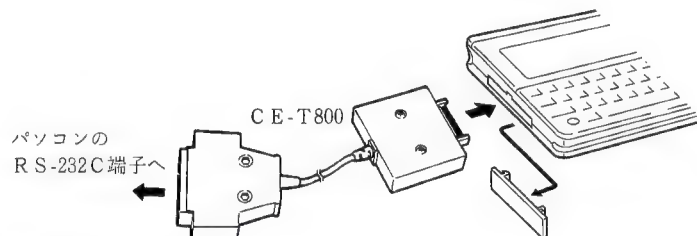
CE-T800は、本機とパソコンなどとの通信（シリアル方式）を行うためのケーブルです。

CE-T800をお持ちの場合、本機とパソコンとの間でTEXTモードのSIO機能を使ったプログラムやデータの入出力、機械語モニタを使った機械語の入出力などができます。

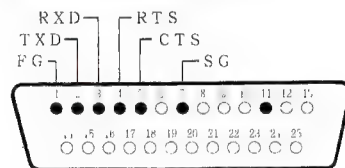
シリアル方式によるパソコンとの通信にはパソコンのRS-232C端子を使用します。本機とRS-232Cでは信号の電圧レベルが異なるため、RS-232Cの信号を直接本機に加えると本機が壊れます。

したがって、この信号レベルを合わせるための回路がCE-T800に入っています。

（注）CE-T800の端子に指などで触れないでください。静電気などにより内部回路が壊れることがあります。



【CE-T800のコネクタ信号〈DB-25(W)〉】



〈図1〉

信号名

| RS-232C信号 |        |         | CE-T800(〈図1〉) |         | 機 能 概 要                                       |
|-----------|--------|---------|---------------|---------|-----------------------------------------------|
| ピン番号      | 信号名    | 記号      | から見た信号の方向     |         |                                               |
| 1         | フレーム接地 | FG      | —             |         |                                               |
| 2         | 送信データ  | TXD(SD) | 入力            | パソコン→本機 | 本機に送信されるデータ信号                                 |
| 3         | 受信データ  | RXD(RD) | 出力(注)         | 本機→パソコン | 本機から送信するデータ信号                                 |
| 4         | 送信要求   | RTS(RS) | 入力            | パソコン→本機 | ハイレベル……パソコンからのデータ送信可<br>ローレベル……パソコンからのデータ送信停止 |
| 5         | 送信可    | CTS(CS) | 出力(注)         | 本機→パソコン | 受信可のとき……ハイレベル<br>受信不可のとき……ローレベル               |
| 7         | 信号用接地  | SG      | —             |         | 入出力装置間の基準電位を合わせます。                            |
| 11        |        | NC      | —             |         | 本機では使用していません。                                 |

（注）・CE-T800は、パソコンのRS-232C端子につないで使用できるように、パソコンからの出力信号（2番、4番）は本機の入力信号に、パソコンの入力信号（3番、5番）へは本機の出力信号が接続されています。

・これらの出力信号は、下記以外の状態では不定です。

①BASICモードおよびC言語モードでSIOがオープンしているとき。

②機械語モニタモードでR命令およびW命令を実行するとき。

③TEXTモードでSIOの送受信をするとき。

## 2. 電池の交換について

電池が消耗して、規定電圧以下になると画面左下に「BATT」シンボルが点灯します。

このシンボルが点灯したときは、「OFF」で電源を切り、再び「ON」で電源を入れてください。それでもこのシンボルが消えないときは、速やかに次の手順で新しい電池と交換してください。このシンボルが点灯した状態で使用し続けると、本機の電源が自動的に切れて何も動作しなくなります。

（注）機械語命令実行中は、電池が消耗しても「BATT」シンボルが点灯しませんので、ご注意ください。

機械語命令を実行したまま長時間放置しますと、電圧が低下し、正常な動作をしなくなる恐れがあります。

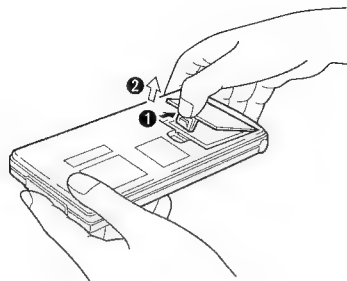
なお、機械語命令実行中に誤動作等が発生した場合は電池の消耗が考えられます。リセットスイッチで実行を止め、「BATT」シンボルの点灯または電源が切れる場合は速やかに電池を交換してください。

### 電池を交換する前に

本機内に大切なプログラムやデータが入っている場合、そのまま電池の交換を行うと、その内容が消えてしまいます。電池を交換する前に、紙に書き写しておいてください。CE-126P（プリンタ）をお持ちの場合は、印字しておいてください。

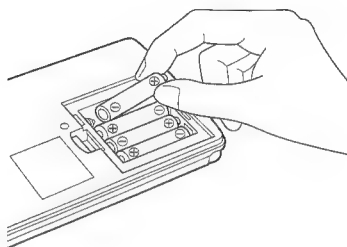
## 電池の交換

①本体裏面の電池ぶたを図のようにして外します。



②乾電池を新しいものと交換します。(乾電池は4本とも交換してください。)

乾電池の⊕・○をまちがえないように、⊖(マイナス)側から入れます。



③電池ぶたをもとどおり取り付けます。

④14ページの「お買いあげ後はじめてご使用になるときの操作」の(2)～(3)の項目を実行します。

⑤パソコンなどに記録しておいた内容を読み込みます。

■乾電池は次のタイプをお使いください。

単4形乾電池R03 4本(指定している電池以外は使用しないでください。)

### ⚠ 電池使用上のご注意

乾電池は誤った使いかたをすると、破れつや発火の原因になることがあります。また、液もれして機器を腐食させたり、手や衣服などを汚す原因になることもあります。以下のことをお守りください。

- 乾電池のプラス(+)とマイナス(-)の向きを本体の表示どおり正しく入れてください。
  - 指定されていない電池を使用しないでください。
  - 使えなくなった電池を本体の中に入れたままにしないでください。
  - 種類の違うものや、新しいものと古いものを混ぜて使用しないでください。
  - もれた液が体についたときは、水でよく洗い流してください。
  - 充電や分解、ショートする恐れがあることはしないでください。
- また、加熱したり、水や火の中に入れたりしないでください。

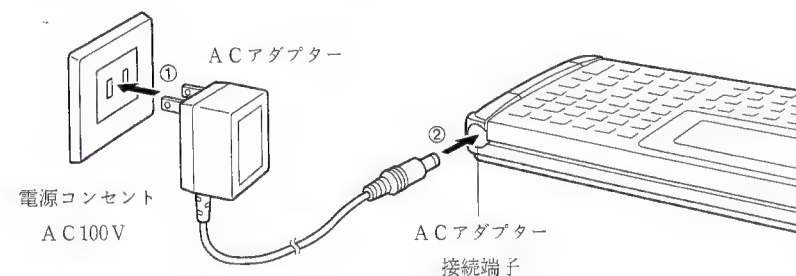


### 使用時間について

付属の電池は工場出荷時に入れていますので、所定の連続使用時間に満たないうちに、寿命が切れることがあります。

## ACアダプターの接続のしかた

別売のACアダプターEA-23Eを使用すると、家庭用電源“AC100V”で動作します。  
本機の電源を切ってから、矢印①、②の順に接続してください。



### ⚠ ご注意

- 本体の電池を取り外したまま、ACアダプターだけで動作させないでください。ACアダプターだけで動作させると、誤って接続プラグが外れた場合、せっかく記憶させたデータがすべて消えてしまいます。このため、ACアダプターは本体の電池が消耗しているときは使用できない(動作しない)ことがあります。
- 通信中にACアダプターの抜き差し(コンセントやACアダプター接続端子から)をしないでください。通信が中断されることがあります。

### ⚠ ACアダプター使用上のご注意

- ACアダプターEA-23E以外のACアダプターを使用しないでください。故障の原因になります。
- ACアダプターEA-23Eを他の機器に使用しないでください。その機器を壊す恐れがあります。

### 3. 主なキーの主な機能

次に主なキーの主な機能を説明します。

| キー                                  | 機 能                                                                                                                                                                                                                                                             |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>BREAK</b><br><b>(ON)</b>         | <ul style="list-style-type: none"> <li>● 電源を入れるときに押します。</li> <li>● プログラム実行中、プログラムを一時停止状態（BREAK：ブレイク）にします。</li> <li>● BSAVEやLOADなどの命令実行中は、その実行を中止します。</li> <li>● 統計計算では、機能の選択画面に戻すときにも使用します。</li> <li>● TEXTモード、C言語機能モードでは、メニュー画面、機能選択画面に戻すときにも使用します。</li> </ul> |
| <b>(OFF)</b>                        | ● 電源を切ります。                                                                                                                                                                                                                                                      |
| <b>(BASIC)</b>                      | <ul style="list-style-type: none"> <li>● RUNモードとPROモードを切り替えます。</li> <li>● 他のモードからRUNモードにします。</li> </ul>                                                                                                                                                         |
| <b>(SHIFT) + (ASMBL)</b>            | ● アセンブラ、CASLまたはPICモードにします。                                                                                                                                                                                                                                      |
| <b>(TEXT)</b>                       | <ul style="list-style-type: none"> <li>● TEXTモードにします。</li> <li>● TEXTの機能選択画面（メインメニュー画面）にします。</li> </ul>                                                                                                                                                         |
| <b>(SHIFT) + <sup>C</sup>(TEXT)</b> | <ul style="list-style-type: none"> <li>● C言語機能モードにします。</li> <li>● C言語機能のメニュー画面にします。</li> </ul>                                                                                                                                                                  |
| <b>(SHIFT) + (コントラスト)</b>           | ● 表示の濃度調整画面にします。                                                                                                                                                                                                                                                |
| <b>(SHIFT)</b>                      | ● 各キーの橙色で書かれている機能を使うとき、このキーを押したまま、それぞれのキーを押します。（2nd F参照）                                                                                                                                                                                                        |
| <small>小文字</small><br><b>(CAPS)</b> | <ul style="list-style-type: none"> <li>● アルファベットの太文字と小文字の入力モードを切り替えます。（画面右側の“CAPS”シンボルの点灯、消灯を行います。）</li> <li>● カナの入力モードのとき、小さいカナ文字（アイウエオヤユヨツ）を入力したいときに押します。（“小”シンボルの点灯、消灯を行います。）</li> </ul>                                                                     |
| <b>(カナ)</b>                         | ● カナ入力モードの設定、解除を行います。（“カナ”シンボルの点灯、消灯を行います。）                                                                                                                                                                                                                     |
| <b>(TAB)</b>                        | ● カーソルを決められた桁数だけ移動させます。<br>RUN、PROモードでは7桁ずつ移動します。TEXTモードのエディタでは、最初8桁、2回目は6桁、3回目以降は7桁ずつ移動させます。                                                                                                                                                                   |
| <b>▶</b>                            | <ul style="list-style-type: none"> <li>● カーソルを右に移動させます。</li> <li>● プレイバックを行います。</li> <li>● プログラムが表示されているときで、カーソルが表示されていない場合はカーソルを呼び出します。</li> <li>● マニュアル計算などでのエラーを解除します。</li> </ul>                                                                            |
| <b>◀</b>                            | <ul style="list-style-type: none"> <li>● カーソルを左に移動させます。</li> <li>● その他は▶と同じ。</li> </ul>                                                                                                                                                                         |
| <b>(ANS)</b>                        | ● ラストアンサーを呼び出します。                                                                                                                                                                                                                                               |
| <b>(CONST)</b>                      | ● 定数計算の定数と計算命令を設定します。（“CONST”シンボル点灯）<br>2nd F (CONST) (SHIFT) + (CONST) を押すと、設定されている定数を表示します。                                                                                                                                                                   |

| キー                      | 機 能                                                                                                                                                                                                                                                                                                  |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>(INS)</b>            | ● 訂正モードと挿入モードの切り替えを行います。                                                                                                                                                                                                                                                                             |
| <b>(SHIFT) + (DEL)</b>  | ● 現在のカーソル位置の文字を削除します。                                                                                                                                                                                                                                                                                |
| <b>(BS)</b>             | ● カーソル位置の1字前の文字を削除します。                                                                                                                                                                                                                                                                               |
| <b>(2nd F)</b>          | ● 各キーの橙色で書かれている機能を使うとき、各キーを押す前にこのキーを押します。（(SHIFT)参照）                                                                                                                                                                                                                                                 |
| <b>(CLS)</b>            | <ul style="list-style-type: none"> <li>● 入力中の内容や表示をクリア（消去）するときに押します。</li> <li>● エラーを解除します。</li> </ul>                                                                                                                                                                                                |
| <b>(SHIFT) + (CA)</b>   | <ul style="list-style-type: none"> <li>● 計算機の状態を解除します。（クリアオール） <ul style="list-style-type: none"> <li>・ プログラムの実行が一時停止状態にあるとき、実行を中止させます。</li> <li>・ 表示内容などを消去します。</li> <li>・ 表示フォーマット指定（USING指定）を解除します。</li> <li>・ トレースモードを解除（TROFF状態に）します。</li> <li>・ エラーを解除します。</li> </ul> </li> <li>その他</li> </ul> |
| <b>↵</b>                | <ul style="list-style-type: none"> <li>● プログラムの行の終了を指定します。</li> <li>● プログラムを計算機に書き込みます。</li> <li>● マニュアル計算の実行、あるいはBASIC命令などのマニュアル操作による実行を行います。</li> <li>● PRINT命令やINPUT命令で一時停止しているプログラムの再スタートなど、プログラムの再開に使用します。</li> </ul>                                                                           |
| <b>(SHIFT) + (P⇔NP)</b> | ● プリンタが接続され、電源が入っているとき、プリントモードの設定、解除を行います。（“PRINT”シンボルの点灯、消灯を行います。）                                                                                                                                                                                                                                  |

▼ ▲ の動きは、モードの指定および計算機の状態によって変わります。BASICモード（RUN、PRO）では次のようになります。

| モード | 状 態                                              | ▼                         | ▲                          |
|-----|--------------------------------------------------|---------------------------|----------------------------|
| RUN | プログラム実行中                                         | 無効                        | 無効                         |
|     | プログラムの一時停止中、WAITが無限のPRINT命令実行中やINPUT命令実行中のブレイク状態 | 次の行を実行<br>（1行ずつ実行して停止します） | 押している間、実行している行あるいは実行した行を表示 |
|     | プログラム実行時のエラー状態                                   | 無効                        | 押している間、エラーが発生した行を表示        |
|     | トレースモードオン状態                                      | トレースを実行                   | 押している間、実行している行あるいは実行した行を表示 |
| PRO | （RUNモードからPROモードに切り替え、プログラムが表示されていないとき）           |                           |                            |
|     | プログラムの一時停止中                                      | 停止している行を表示                | 同 左                        |
|     | エラー発生後                                           | エラーが発生した行を表示              | 同 左                        |
|     | その他                                              | 先頭行を表示                    | 最終行を表示                     |
| PRO | （プログラム行が表示されているとき）                               |                           |                            |
|     | 次のプログラム行を表示                                      | 1行前の行を表示                  |                            |

## 4. 計算範囲

### 加減乗除算

被演算数、演算数、結果が  $\pm 1 \times 10^{-99} \sim \pm 9.999999999 \times 10^{99}$  および 0

### 関数計算

| 関 数                                                 | 計 算 範 囲                                                                                                                                                                                                                                                                                                                                                           | 備 考                                                            |
|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| SIN $x$<br>COS $x$<br>TAN $x$                       | DEG : $ x  < 1 \times 10^{10}$<br>RAD : $ x  < \frac{\pi}{180} \times 10^{10}$<br>GRAD : $ x  < \frac{10}{9} \times 10^{10}$<br>ただし tan $x$ において次の場合は除く<br>DEG : $ x  = 90(2n-1)$<br>RAD : $ x  = \frac{\pi}{2}(2n-1)$<br>GRAD : $ x  = 100(2n-1)$<br>( $n$ は整数)                                                                                                  |                                                                |
| ASN $x$<br>ACS $x$                                  | $-1 \leq x \leq 1$                                                                                                                                                                                                                                                                                                                                                | $\sin^{-1} x$<br>$\cos^{-1} x$                                 |
| ATN $x$                                             | $ x  < 1 \times 10^{100}$                                                                                                                                                                                                                                                                                                                                         | $\tan^{-1} x$                                                  |
| HSN $x$<br>HCS $x$<br>HTN $x$                       | $-227.9559242 \leq x \leq 230.2585092$                                                                                                                                                                                                                                                                                                                            | $\sinh x$<br>$\cosh x$<br>$\tanh x$                            |
| AHS $x$                                             | $ x  < 1 \times 10^{50}$                                                                                                                                                                                                                                                                                                                                          | $\sinh^{-1} x$                                                 |
| AHC $x$                                             | $1 \leq x < 1 \times 10^{50}$                                                                                                                                                                                                                                                                                                                                     | $\cosh^{-1} x$                                                 |
| AHT $x$                                             | $ x  < 1$                                                                                                                                                                                                                                                                                                                                                         | $\tanh^{-1} x$                                                 |
| LN $x$<br>LOG $x$                                   | $1 \times 10^{-99} \leq x < 1 \times 10^{100}$                                                                                                                                                                                                                                                                                                                    | $\ln x = \log_e x$                                             |
| EXP $x$                                             | $-1 \times 10^{100} < x \leq 230.2585092$                                                                                                                                                                                                                                                                                                                         | $e^x$ $e \approx 2.718281828$                                  |
| TEN $x$                                             | $-1 \times 10^{100} < x < 100$                                                                                                                                                                                                                                                                                                                                    | $10^x$                                                         |
| RCP $x$<br>SQU $x$<br>CUB $x$<br>SQR $x$<br>CUR $x$ | $ x  < 1 \times 10^{100}$ $x \neq 0$<br>$ x  < 1 \times 10^{50}$<br>$ x  < 2.154434690 \times 10^{33}$<br>$0 \leq x < 1 \times 10^{100}$<br>$ x  < 1 \times 10^{100}$                                                                                                                                                                                             | $\frac{1}{x}$<br>$x^2$<br>$x^3$<br>$\sqrt{x}$<br>$\sqrt[3]{x}$ |
| $y \wedge x$ ( $y^x$ )                              | <ul style="list-style-type: none"> <li>● <math>y &gt; 0</math> のとき<br/><math>-1 \times 10^{100} &lt; x \log y &lt; 100</math></li> <li>● <math>y = 0</math> のとき<br/><math>x &gt; 0</math></li> <li>● <math>y &lt; 0</math> のとき<br/><math>x</math> は整数または <math>\frac{1}{x}</math> が奇数<br/>ただし <math>-1 \times 10^{100} &lt; x \log  y  &lt; 100</math></li> </ul> | $y^x = 10^{x \cdot \log y}$                                    |

| 関 数                                                     | 計 算 範 囲                                                                                                                                                  | 備 考                                                        |
|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
| &H $x$                                                  | $0 \leq x \leq 2540\text{BE}3\text{FF}$ または<br>$\text{FDABF41C01} \leq x \leq \text{FFFFFFFFFFFF}$                                                       | $x$ は16進数での整数                                              |
| POL ( $x, y$ )<br>( $x, y \rightarrow r, \theta$ )      | $(x^2 + y^2) < 1 \times 10^{100}$<br>$\frac{x}{y} < 1 \times 10^{100}$                                                                                   | $r = \sqrt{x^2 + y^2}$<br>$\theta = \tan^{-1} \frac{y}{x}$ |
| REC ( $r, \theta$ )<br>( $r, \theta \rightarrow x, y$ ) | $r < 1 \times 10^{100}$<br>$\theta$ の範囲は $\sin x, \cos x$ の $x$ と同じ                                                                                      | $x = r \cos \theta$<br>$y = r \sin \theta$                 |
| NPR ( $n, r$ )                                          | $\frac{n!}{(n-r)!} < 1 \times 10^{100}$ $0 \leq r \leq n \leq 9999999999$<br>$r, n$ は整数                                                                  | $nPr$                                                      |
| NCR ( $n, r$ )                                          | $\frac{n!}{(n-r)!r!} < 1 \times 10^{100}$ $0 \leq r \leq n \leq 9999999999$<br>$r, n$ は整数<br>$n-r < r$ のとき $n-r \leq 69$<br>$n-r \geq r$ のとき $r \leq 69$ | $nCr$                                                      |
| FACT $x$                                                | $0 \leq x \leq 69$                                                                                                                                       | $n!$                                                       |
| DEG $x$                                                 | $ x  < 1 \times 10^4$                                                                                                                                    | DMS $\rightarrow$ DEG                                      |
| DMS $x$                                                 | $ x  < 1 \times 10^4$                                                                                                                                    | DEG $\rightarrow$ DMS                                      |

### 統計計算

|     | 計 算 範 囲                                                                                       |                                |                                                       |              |
|-----|-----------------------------------------------------------------------------------------------|--------------------------------|-------------------------------------------------------|--------------|
| データ | $ x  < 1 \times 10^{50}$                                                                      | $1 \leq n < 1 \times 10^{100}$ | $ y  < 1 \times 10^{50}$                              |              |
| 統計量 | 次の計算において、計算結果および途中結果の絶対値が $1 \times 10^{100}$ 未満であること。<br>分母 (除数) が 0 でないこと。√ で計算する値が負数でないこと。 |                                |                                                       |              |
|     | $\Sigma x$                                                                                    | $\Sigma x^2$                   | $\Sigma y$                                            | $\Sigma y^2$ |
|     | $\bar{x} = \frac{\Sigma x}{n}$                                                                |                                | $\bar{y} = \frac{\Sigma y}{n}$                        |              |
|     | $sx = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n-1}}$                                             |                                | $sy = \sqrt{\frac{\Sigma y^2 - n\bar{y}^2}{n-1}}$     |              |
|     | $\sigma x = \sqrt{\frac{\Sigma x^2 - n\bar{x}^2}{n}}$                                         |                                | $\sigma y = \sqrt{\frac{\Sigma y^2 - n\bar{y}^2}{n}}$ |              |
|     | $a = \bar{y} - b\bar{x}$                                                                      |                                | $b = \frac{Sxy}{Sxx}$                                 |              |
|     | $r = \frac{Sxy}{\sqrt{Sxx \cdot Syy}}$                                                        |                                | $Sxx = \Sigma x^2 - \frac{(\Sigma x)^2}{n}$           |              |
|     | $x' = \frac{y-a}{b}$                                                                          |                                | $Syy = \Sigma y^2 - \frac{(\Sigma y)^2}{n}$           |              |
|     | $y' = a + bx$                                                                                 |                                | $Sxy = \Sigma xy - \frac{\Sigma x \cdot \Sigma y}{n}$ |              |

計算の誤差は原則として、10桁目に  $\pm 1$  となります。(指数表示の場合は仮数部表示の最下位桁に  $\pm 1$  となります。)

ただし、関数の特異点および変曲点の近くでは誤差が累積されて大きくなります。また、連続計算を行った場合もそれぞれの誤差が累積されて大きくなります。(べき乗 ( $y^x$ ) のように、計算機内で連続計算を行っている場合も同様です。)



## 5. 仕様

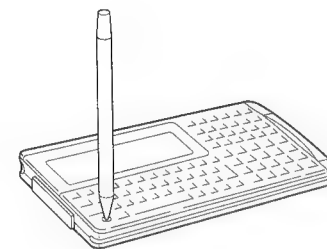
|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 形 名             | PC-G850VS                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| C P U           | CMOS 8ビットCPU (Z80相当品)                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| メ モ リ 容 量       | システムエリア……………約2.3Kバイト<br>データ専用エリア……………208バイト<br>プログラム・データエリア……………30179バイト                                                                                                                                                                                                                                                                                                                                                                                                     |
| ス タ ッ ク         | ファンクション用 16段      データ用 8段      サブルーチン用 10段<br>構造化BASIC用      合計90バイト                                                                                                                                                                                                                                                                                                                                                                                                         |
|                 | <div style="display: flex; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 5px; margin-right: 5px;"> REPEAT~UNTIL: 1段で4バイト<br/> WHILE~WEND: 1段で5バイト<br/> SWITCH~CASE: 1段で6バイト<br/> FOR~NEXT: 1段で18バイト </div> <div> (ただし、SWITCH~CASEは1段しか使用できません。) </div> </div>                                                                                                                                                                               |
| 基本計算機能          | 基本計算: 加減乗除算<br>関数計算: 三角関数、逆三角関数、対数、指数、角度変換、べき乗、開平計算、<br>整数化、絶対値、符号関数、円周率、座標変換、その他                                                                                                                                                                                                                                                                                                                                                                                            |
| 計 算 桁 数         | 10桁 (仮数部) + 2桁 (指数部)                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 計 算 方 法         | 数式どおり (優先順位判別機能つき)                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 周辺機器接続端子 (11ピン) | CE-126P (プリンタ)、CE-T800 (パソコン通信ケーブル)、EA-129C (ポケ<br>コン接続ケーブル)                                                                                                                                                                                                                                                                                                                                                                                                                 |
| システムバス端子 (40ピン) |                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 表 示             | 液晶表示 5×7ドットマトリックス表示 (24桁×6行)                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 使 用 温 度         | 0℃~40℃                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 電 源             | 6V… (DC): 単4形乾電池R03 4本<br>(AC100V 50/60Hz: 別売のACアダプターEA-23E使用)                                                                                                                                                                                                                                                                                                                                                                                                               |
| 電池使用時間          | 実使用状態で連続使用 約90時間 (単4形乾電池R03の場合)<br><div style="border-left: 1px solid black; padding-left: 5px; margin-left: 20px;"> 使用温度25℃で1時間当り演算またはプログラム実行を10分間、表示状態を<br/> 50分間行った場合 </div> (注) パソコン通信ケーブルCE-T800を使用して通信を行っているときの電池使用<br>時間は約70時間になります。<br><div style="border-left: 1px solid black; padding-left: 5px; margin-left: 20px;"> 使用温度25℃で、通信を2分間、演算またはプログラム実行を8分間、<br/> 表示状態を50分間行った場合 </div> <ul style="list-style-type: none"> <li>● 電池の種類や使用方法などにより多少の変動があります。</li> </ul> |
| 消 費 電 力         | 約0.2W                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 外 形 寸 法         | 幅196mm×奥行95mm×厚さ20mm                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 質 量             | 約260g (乾電池含む。ハードカバーは除く。)                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 付 属 品           | ハードカバー、単4形乾電池 4本、ネームラベル (本体裏面に貼りつけ済み)、<br>取扱説明書※、お客様ご相談窓口のご案内<br>※当商品は日本国内向けであり、日本語以外の説明書はございません。<br>This model is designed exclusively for Japan, with manuals in Japanese only.                                                                                                                                                                                                                                                                                            |

## 異常が発生した場合の処理

BREAK  
**ON** を含めたすべてのキーの機能が働かない、あるいは正しく動作しないなどの異常が発生した場合は、  
 本機の電源を入れたままで周辺機器の接続や取り外しを行ったか、またはプログラムのミス、あるいは強  
 度の外来ノイズなどによる異常発生などの原因が考えられます。  
 または、電池交換を行った後、リセットスイッチを押さなかった場合が考えられます。  
 このような場合は、次の方法でリセットしてください。

### リセットのしかた

- ① **ON** を押して電源を入れた後、ボールペンなどで、  
 本体左端のリセット (RESET) スイッチを押し  
 てください。  
 芯先の出たシャープペンシルや先の折れやすいもの、  
 また、針など先のとがったものは使用しないでくだ  
 さい。



- ② リセットスイッチを離すと次の画面になります。違う  
 画面になったときはもう一度リセットスイッチを押し  
 てください。

MEMORY CLEAR O. K. ? (Y/N)

(メモリー内容を消去しますか?)

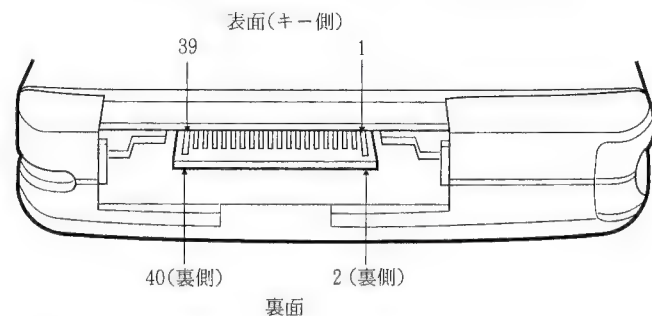
- ③ プログラムやデータを保持したいときは、  
**N** を押してください。RUNモードの最初の画面になります。  
 ● この後、プログラム実行などで再び異常が発生する場合は、次のプログラムやデータなどをすべて  
 消去する方法を行ってください。

プログラムやデータなどをすべて消去するときは、  
 上記の表示中に **Y** を押してください。記憶内容を  
 すべて消去して、次の画面 (点滅) になります。  
 (初期設定し、記憶内容をすべて消去したことを示  
 しています。)



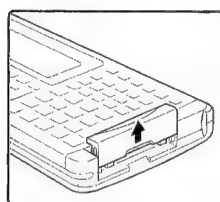
この画面で、**R** を押せばRUNモードの最初の画  
 面になります。

# システムバス端子信号表



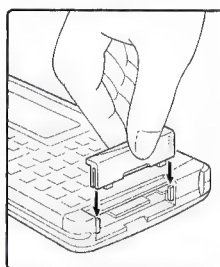
| 表 面    |      | 裏 面  |         |
|--------|------|------|---------|
| 信号名    | 端子番号 | 端子番号 | 信号名     |
| Vcc    | 1    | 2    | Vcc     |
| M1     | 3    | 4    | MREQ    |
| IORQ   | 5    | 6    | IORESET |
| WAIT   | 7    | 8    | INT1    |
| WR     | 9    | 10   | RD      |
| BNK1   | 11   | 12   | BNK0    |
| CEROM2 | 13   | 14   | CERAM2  |
| D7     | 15   | 16   | D6      |
| D5     | 17   | 18   | D4      |
| D3     | 19   | 20   | D2      |
| D1     | 21   | 22   | D0      |
| A15    | 23   | 24   | A14     |
| A13    | 25   | 26   | A12     |
| A11    | 27   | 28   | A10     |
| A9     | 29   | 30   | A8      |
| A7     | 31   | 32   | A6      |
| A5     | 33   | 34   | A4      |
| A3     | 35   | 36   | A2      |
| A1     | 37   | 38   | A0      |
| GND    | 39   | 40   | GND     |

システムバス端子カバー  
を取り外すとき



上にスライドさせる。

システムバス端子カバー  
を取り付けるとき



みぞにあわせてスライド  
させる。

(注) 本機の Vcc 電源電圧は、電池の消耗度合いにより、4～6 V の間で変動します。

本機内部は、C-MOS 部品により構成されているため、端子に Vcc + 0.5V 以上の信号電圧が加えられると、本機内部が壊れることがあります。

# ローマ字→カナ変換表

|    | A   |                 | I   |    | U   |    | E   |    | O   |    |
|----|-----|-----------------|-----|----|-----|----|-----|----|-----|----|
| ア行 | A   | ア               | I   | イ  | U   | ウ  | E   | エ  | O   | オ  |
|    |     |                 |     |    |     |    | YE  | イエ |     |    |
| カ行 | KA  | カ               | KI  | キ  | KU  | ク  | KE  | ケ  | KO  | コ  |
|    | CA  | カ               |     |    | CU  | ク  |     |    | CO  | コ  |
|    | QA  | クァ              | QI  | クィ | QU  | ク  | QE  | クエ | QO  | クォ |
|    | KYA | キャ              | KYI | キィ | KYU | キュ | KYE | キエ | KYO | キョ |
| サ行 | SA  | サ               | SI  | シ  | SU  | ス  | SE  | セ  | SO  | ソ  |
|    | SHA | シャ              | SHI | シ  | SHU | シュ | SHE | シェ | SHO | ショ |
|    | SYA | シャ              | SYI | シィ | SYU | シュ | SYE | シェ | SYO | ショ |
| タ行 | TA  | タ               | TI  | チ  | TU  | ツ  | TE  | テ  | TO  | ト  |
|    | TSA | ツァ              | TSI | ツイ | TSU | ツ  | TSE | ツエ | TSO | ツォ |
|    | CHA | チャ              | CHI | チ  | CHU | チュ | CHE | チェ | CHO | チョ |
|    | TYA | チャ              | TYI | チィ | TYU | チュ | TYE | チェ | TYO | チョ |
|    | CYA | チャ              | CYI | チィ | CYU | チュ | CYE | チェ | CYO | チョ |
| ナ行 | NA  | ナ               | NI  | ニ  | NU  | ヌ  | NE  | ネ  | NO  | ノ  |
|    | NYA | ニャ              | NYI | ニィ | NYU | ニュ | NYE | ニエ | NYO | ニョ |
| ハ行 | HA  | ハ               | HI  | ヒ  | HU  | フ  | HE  | ヘ  | HO  | ホ  |
|    | FA  | ファ              | FI  | フィ | FU  | フ  | FE  | フェ | FO  | フォ |
|    | HYA | ヒャ              | HYI | ヒィ | HYU | ヒュ | HYE | ヒエ | HYO | ヒョ |
| マ行 | MA  | マ               | MI  | ミ  | MU  | ム  | ME  | メ  | MO  | モ  |
|    | MYA | ミャ              | MYI | ミィ | MYU | ミュ | MYE | ミエ | MYO | ミョ |
| ヤ行 | YA  | ヤ               | YI  | イ  | YU  | ユ  |     |    | YO  | ヨ  |
| ラ行 | RA  | ラ               | RI  | リ  | RU  | ル  | RE  | レ  | RO  | ロ  |
|    | LA  | ラ               | LI  | リ  | LU  | ル  | LE  | レ  | LO  | ロ  |
|    | RYA | リャ              | RYI | リィ | RYU | リュ | RYE | リエ | RYO | リョ |
|    | LYA | リャ              | LYI | リィ | LYU | リュ | LYE | リエ | LYO | リョ |
| ワ行 | WA  | ワ               |     |    |     |    |     |    | WO  | ヲ  |
| ン  | N   | N (SHIFT) + (U) |     | M  |     |    |     |    |     |    |

|    | A   |    | I   |    | U   |    | E   |    | O   |    |
|----|-----|----|-----|----|-----|----|-----|----|-----|----|
| ア行 | VA  | ヴァ | VI  | ヴィ | VU  | ヴ  | VE  | ヴェ | VO  | ヴォ |
| ガ行 | GA  | ガ  | GI  | ギ  | GU  | グ  | GE  | ゲ  | GO  | ゴ  |
|    | GYA | ギャ | GYI | ギィ | GYU | ギュ | GYE | ギェ | GYO | ギョ |
| ザ行 | ZA  | ザ  | ZI  | ジ  | ZU  | ズ  | ZE  | ゼ  | ZO  | ゾ  |
|    | JA  | ジャ | JI  | ジ  | JU  | ジュ | JE  | ジェ | JO  | ジョ |
|    | JYA | ジャ | JYI | ジィ | JYU | ジュ | JYE | ジェ | JYO | ジョ |
|    | ZYA | ジャ | ZYI | ジィ | ZYU | ジュ | ZYE | ジェ | ZYO | ジョ |
| ダ行 | DA  | ダ  | DI  | ヂ  | DU  | ヅ  | DE  | デ  | DO  | ド  |
|    | DHA | デハ | DHI | ディ | DHU | デュ | DHE | デュ | DHO | ドホ |
|    | DYA | チャ | DYI | ディ | DYU | ヂュ | DYE | ヂュ | DYO | ヂョ |
| バ行 | BA  | バ  | BI  | ビ  | BU  | ブ  | BE  | ベ  | BO  | ボ  |
|    | BYA | ビヤ | BYI | ビィ | BYU | ビュ | BYE | ビェ | BYO | ビョ |
| パ行 | PA  | パ  | PI  | ピ  | PU  | プ  | PE  | ペ  | PO  | ポ  |
|    | PYA | ピヤ | PYI | ピィ | PYU | ピュ | PYE | ピェ | PYO | ピョ |

## ①「ン」の入力

「ン」は「N」と入力します。

ただし、「N」の次に母音（A、I、U、E、O）および「Y」がくるとき、または後に文字がこないときは「N **SHIFT** + **U**」と押します。

なお、「N」の代わりに「M」を用いることもできます。

（例） シンユウ → SIN **SHIFT** + **U** YUU

シンニユウ → SINNYUU

シンブン → SIMBUN **SHIFT** + **U**

## ②「ッ」の入力

「ッ」は子音を重ねて入力します。ただし、「N」や「M」を重ねて入力したときは「ン」が入ります。

（例） キップ → KIPPU

セット → SETTO

## ③小さい文字の単独入力

小さい文字（ア、イ、ウ、エ、オ、ツ、ヤ、ユ、ヨ）を入力するときは 小文字 **CAPS** を押してからそれぞれの文字を入力します。

（例） ティーカップ → TE 小文字 **CAPS** I - KAPPU

## キャラクタ・コード表

| 下<br>位<br>桁 | 上位桁  | 0  | 16 | 32   | 48 | 64 | 80 | 96 | 112 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |
|-------------|------|----|----|------|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
|             | 16進数 | 0  | 1  | 2    | 3  | 4  | 5  | 6  | 7   | 8   | 9   | A   | B   | C   | D   | E   | F   |
| 0           | 0    | ヌル |    | スペース | 0  | @  | P  | '  | p   |     | ー   |     | ー   | タ   | ミ   | =   | ×   |
| 1           | 1    |    |    | !    | 1  | A  | Q  | a  | q   | ー   | ー   | 。   | ア   | チ   | ム   | ト   | 円   |
| 2           | 2    |    |    | "    | 2  | B  | R  | b  | r   | ー   | ー   | ー   | イ   | ツ   | メ   | キ   | 年   |
| 3           | 3    |    |    | #    | 3  | C  | S  | c  | s   | ー   | ー   | ー   | ウ   | テ   | モ   | コ   | 月   |
| 4           | 4    |    |    | \$   | 4  | D  | T  | d  | t   | ー   | ー   | 、   | エ   | ト   | ヤ   | ノ   | 日   |
| 5           | 5    |    |    | %    | 5  | E  | U  | e  | u   | ー   | ー   | ・   | オ   | ナ   | ユ   | ノ   | 時   |
| 6           | 6    |    |    | &    | 6  | F  | V  | f  | v   | ー   | ー   | ラ   | カ   | ニ   | ヨ   | ノ   | 分   |
| 7           | 7    |    |    | '    | 7  | G  | W  | g  | w   | ー   | ー   | ア   | キ   | ヌ   | ラ   | ノ   | 秒   |
| 8           | 8    |    |    | (    | 8  | H  | X  | h  | x   | ー   | ー   | イ   | ク   | ネ   | リ   | ノ   | "   |
| 9           | 9    |    |    | )    | 9  | I  | Y  | i  | y   | ー   | ー   | ウ   | ケ   | ノ   | ル   | ノ   |     |
| 10          | A    |    |    | *    | :  | J  | Z  | j  | z   | ー   | ー   | エ   | コ   | ハ   | レ   | ノ   |     |
| 11          | B    |    |    | +    | ;  | K  | [  | k  | {   | ー   | ー   | オ   | サ   | ヒ   | ロ   | ノ   |     |
| 12          | C    |    |    | ,    | <  | L  | ¥  | l  | :   | ー   | ー   | ヤ   | シ   | フ   | ワ   | ノ   |     |
| 13          | D    |    |    | -    | =  | M  | ]  | m  | }   | ー   | ー   | ユ   | ス   | ヘ   | ン   | ノ   |     |
| 14          | E    |    |    | .    | >  | N  | ^  | n  | ~   | ー   | ー   | ヨ   | セ   | ホ   | ノ   | ノ   |     |
| 15          | F    |    |    | /    | ?  | O  | _  | o  |     | ー   | ー   | ッ   | ソ   | マ   | ノ   | ノ   |     |

キャラクタコードは次のように表します。

〈例〉 “\*” のコード

16進数 &H 2 A

10進数 42 (32+10)

〔補足〕

CHR \$ 命令により、本機でキャラクタを表示させる場合

● 表中のコード 0 (&H 0 0) のキャラクタはヌル (Null: 何もない状態) です。

したがって何も表示されません。

● キャラクタが記載されていない部分はスペース (空) になります。

CHR \$ 命令により、CE-126 P でキャラクタを印字させる場合

● キャラクタが記載されていない部分はスペース (空) になります。

● 次のコードはスペースになります。

129(&H 8 1)~159(&H 9 F)、224(&H E 0)~231(&H E 7)、236(&H E C)~240(&H F 0)、  
245(&H F 5)~248(&H F 8)

## メモリマップ・I/Oマップ

### メモリマップ

|        |                                 |               |               |
|--------|---------------------------------|---------------|---------------|
| 0000H  | リザーブエリア                         | 使用不可          | 00H           |
| 0100H  | 機械語エリア                          | システムポート       | 10H           |
|        | プログラムファイルエリア                    | 空きポート         | 20H           |
|        | ラムデータファイルエリア                    | 空きポート         | 30H           |
|        | テキストエリア                         | ディスプレイ用       | 40H           |
|        | BASICプログラムエリア<br>(プログラム・データエリア) | ディスプレイ用       | 50H           |
|        |                                 | システムポート       | 60H           |
|        |                                 | システムポート       | 70H           |
|        |                                 |               | 80H           |
|        | 変数エリア                           | 使用不可          |               |
|        | ワークエリア                          |               |               |
|        | 固定変数エリア                         |               |               |
|        | ワークエリア                          |               |               |
|        | スタックエリア                         |               |               |
| 8000H  | ROM BANK 0                      |               | FFH           |
|        |                                 |               |               |
|        |                                 |               |               |
|        |                                 |               |               |
|        |                                 |               |               |
| C 000H | ROM BANK 1                      | ROM<br>BANK 2 | ROM<br>BANK 3 |
|        |                                 |               |               |
|        |                                 |               |               |
|        |                                 |               |               |
|        |                                 |               |               |
| FFFFH  |                                 |               |               |

### I/Oマップ

## エラーコード表

| エラーコード | 内 容                                                                                                         |
|--------|-------------------------------------------------------------------------------------------------------------|
| 10     | 文法的に実行できない場合。                                                                                               |
| 12     | PROモードでしか使えない命令 (LIST、RENUMなど) をRUNモードで使おうとした。または、逆にRUNモードでしか使えない命令をPROモードで使おうとした。<br>OPEN命令のモード指定がまちがっている。 |
| 13     | CONT命令でプログラムの再実行ができない。                                                                                      |
| 14     | BASICプログラムがないときに、パスワードを設定しようとした。                                                                            |
| 15     | BSAVE M命令で、アドレスの指定が逆。(開始アドレスよりも、終了アドレスの方が小さくなっている。)                                                         |
| 20     | 計算結果が $1 \times 10^{100}$ 以上になった。                                                                           |
| 21     | 除数が0の除算を実行した。                                                                                               |
| 22     | 不合理な演算を行った。(例 LOG-3)<br>計算範囲外の値の関数計算を行った。(例 ASN 2)                                                          |
| 30     | すでに宣言されている配列変数名を再度宣言している。                                                                                   |
| 31     | DIM命令で宣言していない配列変数を使用している。                                                                                   |
| 32     | 配列変数の添字がDIM命令で宣言した大きさを超えている。                                                                                |
| 33     | 指定している数値が規定の範囲から外れている。                                                                                      |
| 40     | 指定した行番号やラベルが存在しない。                                                                                          |
| 41     | 行番号 (ラインナンバー) が1～65279の範囲外になっている。                                                                           |
| 43     | RENUM、LCOPY命令の指定に不都合がある。<br>(指定した開始行以降の行番号のつけ替えを行うと、開始行よりも小さい番号の行 (行番号のつけ替えをしない行) と混ざってしまうような指定になっている。)     |
| 44     | LLIST、DELETEなどの命令で指定した開始行と終了行の大きさが逆になっている。(終了行よりも開始行が大きくなっている。)                                             |
| 50     | GOSUB、FOR、REPEAT、WHILEもしくはSWITCHでの段数オーバー。                                                                   |
| 51     | RETURN文に対するGOSUB文がない。                                                                                       |
| 52     | NEXT文に対するFOR文がない。                                                                                           |
| 53     | READ文に対するDATA文がない。                                                                                          |
| 54     | ファンクションバッファ (16段) または、データバッファ (8段) の段数オーバー。                                                                 |
| 55     | 文字記憶エリア (255文字) の容量を超えた。<br>1行の長さが255バイトを超えた。                                                               |
| 60     | プログラムおよび変数が大きすぎて、プログラム・データエリアの容量を超えた。                                                                       |
| 61     | ブロック形式のENDIFがないのにブロック形式のIF文もしくはブロック形式のELSE文を実行した。                                                           |
| 62     | UNTILに対するREPEATがない。                                                                                         |
| 63     | WHILEに対するWENDがない。                                                                                           |



| エラーコード | 内 容                                                                      |
|--------|--------------------------------------------------------------------------|
| 64     | WENDに対するWHILEがない。                                                        |
| 66     | DEFAULTを実行中にCASEまたはDEFAULTを実行しようとした。                                     |
| 68     | SWITCH、CASEもしくはDEFAULTに対するENDSWITCHがない。                                  |
| 69     | SWITCHが実行されていないのにCASE、DEFAULTまたはENDSWITCHを実行した。                          |
| 70     | USING命令で指定されたフォーマットで表示または印字できない。                                         |
| 71     | USING命令の指定が正しくない。(「」で囲んで指定した内容が正しくない。)                                   |
| 72     | 入出力装置に関するエラー。                                                            |
| 77     | ファイルの容量が足りない。                                                            |
| 80     | SIOに対するリードイン(読み込み)エラー。                                                   |
| 81     | SIO、ミニI/Oなどのタイムアウトエラー(プログラムやデータの入出力で、規定の待ち時間を越えた。)                       |
| 82     | BLOAD ?命令による内容の照合で、内容の不一致がある。                                            |
| 83     | INPUT #命令で指定している変数の型が、読み込もうとしているデータの型と一致していない。                           |
| 84     | プリンタ関連のエラー。                                                              |
| 85     | PRINT #、INPUT #命令でデータの入出力を行うとき、相手の装置(デバイス)がオープンされていない。                   |
| 86     | 1つの装置(デバイス)がオープンしているときに、同じファイル番号でほかの装置をオープンしようとした。または、ファイルがすでにオープンされている。 |
| 87     | ファイルのデータを最後まで読み込んだ後、さらにデータを読み込もうとした。                                     |
| 90     | 数値変数に文字、文字変数に数値を代入しようとした。また、SIN A\$のように数値を扱う関数に文字変数を指定したなど、変数名の不適合。      |
| 91     | 固定変数において、数値が入っている変数を文字変数として使用したり、文字が入っている変数を数値変数として使用した。                 |
| 92     | パスワードが一致していない。                                                           |
| 93     | パスワードが設定されているときに、マニュアルでMON、PEEK、POKE、RENUMなどの命令を実行しようとした。                |
| 94     | 指定されたファイルが存在しない。                                                         |
| 95     | ファイル名の指定が正しくない。                                                          |
| 96     | BASICモードでTEXTファイルを読み込もうとした。                                              |
| 97     | ファイルの数が255を超えた。                                                          |

C言語でのコンパイル時と実行時のエラーメッセージについては241ページを参照してください。

## 故障かな?と思ったら

次のような場合は故障でないことがありますので、修理を依頼される前にもう一度お確かめください。それでも具合の悪いときは、次ページの「アフターサービスについて」をご覧のうえ修理を依頼してください。

| こんなとき                     | ここをお確かめください                                                                                                                                                 |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 電源が入らない                   | <ul style="list-style-type: none"> <li>● 新しい電池と交換してください。(☞372ページ)</li> <li>● リセットしてください。(☞380ページ)</li> </ul>                                                |
| 表示が薄いまたは濃い                | <ul style="list-style-type: none"> <li>● 表示濃度を調整してください。(☞17ページ)</li> <li>● “<b>BATT</b>” が点灯しているときは、電池を交換してください。(☞372ページ)</li> </ul>                        |
| すべてのキーが働かない<br>または正しく働かない | <ul style="list-style-type: none"> <li>● リセットしてください。(☞380ページ)</li> </ul>                                                                                    |
| プログラムが表示できない              | <ul style="list-style-type: none"> <li>● パスワードを解除してください(PASS命令でプログラムの表示を禁止しているとき)。(☞348ページ)</li> <li>● 最初からプログラムを入れ直してください(何かの原因でプログラムが消えているとき)。</li> </ul> |
| プリンタで印字できない               | <ul style="list-style-type: none"> <li>● RUNモードで、POKE &amp;H7774, 0を実行してください(メモリーの一部が、本機の内部制御コードと偶然一致したとき)。</li> </ul>                                      |

## アフターサービスについて

### 保証について

1. この製品には取扱説明書の巻末に保証書がついています。  
保証書は販売店にて所定事項を記入してお渡しいたしますので、内容をよくお読みのうえ大切に保存してください。
2. 保証期間はお買いあげの日から3年間です。  
保証期間中でも有料になることがありますので、保証書をよくお読みください。
3. 保証期間後の修理は……  
修理によって機能が維持できる場合は、ご要望により有料修理いたします。

### 補修用性能部品の保有期間

- ・当社はポケットコンピュータの補修用性能部品を製品の製造打切後7年保有しています。
- ・補修用性能部品とは、その製品の機能を維持するために必要な部品です。

### 修理を依頼されるときは

1. 異常があるときは使用をやめて、お買いあげの販売店にこの製品を **お持込み** のうえ、修理をお申しつけください。ご自分での修理はしないでください。
2. アフターサービスについてわからないことは……  
お買いあげの販売店、またはもよりのシャープお客様ご相談窓口にお問い合わせください。
3. ポケットコンピュータに接続した周辺機器に起因して、ポケットコンピュータ本体が故障・損傷した場合は、保証書に記載されている無料修理規定が適用されません（有料修理となります）ので、あらかじめご了承ください。

### お問い合わせは

この製品についてのご意見、ご質問は、もよりのシャープお客様ご相談窓口へお申しつけください。  
付属の「お客様ご相談窓口のご案内」のとおり、全国にお客様ご相談窓口を設けております。

=メ モ=

=メ モ=

**SHARP**

**ポケットコンピュータ保証書** 持込  
修理  
(WARRANTY CARD)

本書は、記載内容の範囲で無料修理をさせていただくことをお約束するものです。保証期間中に故障が発生した場合は、商品と本書をご持参のうえ、お買いあげの販売店にご依頼し本書をご提示ください。お買いあげ年月日、販売店名など記入もれがあると無効です。記入のない場合は、お買いあげの販売店にお申し出ください。

ご転居・ご贈答品などでお買いあげの販売店に修理をご依頼できない場合は、商品に同梱しております「お客様ご相談窓口のご案内」をご覧ください。お客様ご相談窓口にお問い合わせください。

本書は再発行いたしません。たいせつに保管してください。

**〈無料修理規定〉**

1. 取扱説明書・本体注意ラベルなどの注意書にしたがった正常な使用状態で、保証期間内に故障した場合には、お買いあげの販売店が無料修理いたします。  
なお、故障の内容によりまして、修理にかえ同等製品と交換させていただくことがあります。
2. 保証期間内でも、次の場合には有料修理となります。
  - (イ) 本書のご提示がない場合。
  - (ロ) 本書にお買いあげ年月日・お客様名・販売店名の記入がない場合、または字句を書き換えられた場合。
  - (ハ) 使用上の誤り、または不当な修理や改造による故障・損傷。
  - (ニ) お買いあげ後に落とされた場合などによる故障・損傷。
  - (ホ) 火災・公害および地震・雷・風水害その他天災地変など、外部に要因がある故障・損傷。
  - (ヘ) 消耗部品（乾電池）が損耗し、取り替えが必要な場合。
  - (ト) 電池の液漏れ、または、指定規格外の電池の使用による故障・損傷。
  - (チ) 持込修理の対象商品を直接メーカーへ送付した場合の送料等はおお客様の負担となります。また、出張修理等を行った場合には、出張料はおお客様の負担となります。
3. 本書は日本国内においてのみ有効です。  
(THIS WARRANTY CARD IS ONLY VALID FOR SERVICE IN JAPAN.)

★この保証書は本書に明示した期間・条件のもとにおいて無料修理をお約束するものです。したがってこの保証書によって保証書を発行している者（保証責任者）、および、それ以外の事業者に対するお客様の法律上の権利を制限するものではありませんので、保証期間経過後の修理などにつきましておわかりにならない場合は、お買いあげの販売店またはシャープお客様ご相談窓口にお問い合わせください。

修理メモ

|                |                                                    |     |  |
|----------------|----------------------------------------------------|-----|--|
| 形名             | PC-G850VS                                          |     |  |
| お客様            | ふりがな<br>お名前                                        | 様 ☎ |  |
|                | 〒<br>ご住所                                           |     |  |
| 取扱販売店名・住所・電話番号 |                                                    |     |  |
| 保証期間           | お買いあげ日<br>年    月    日より <b>本体は3年間</b><br>(消耗部品は除く) |     |  |

シャープ株式会社

〒545-8522 大阪市阿倍野区長池町22-22

お問合せ先：お客様相談センター



0120-303-909

フリーダイヤルが使用できない場合のご利用は  
または

043-351-1822  
06-6792-1583



■よくある質問などはパソコン  
から検索できます。



<http://www.sharp.co.jp/support/>

シャープ お問い合わせ **検 索**



使用方法・お買い物相談など



【お客様相談センター】

**0120 - 303 - 909**

携帯電話・PHSからもご利用いただけます。

■IP電話などからフリーダイヤルサービスをご利用いただけない場合は…

|         | 電 話              | ファックス            |
|---------|------------------|------------------|
| 東日本相談室→ | 043 - 351 - 1822 | 043 - 299 - 8280 |
| 西日本相談室→ | 06 - 6792 - 1583 | 06 - 6792 - 5993 |

受付時間 ●月曜～土曜:9:00～18:00 ●日曜・祝日:9:00～17:00 (年末年始を除く)



修理のご相談など



【修理相談センター】(沖縄・奄美地区を除く)

**0570 - 02 - 4649**

全国どこからでも一律料金でご利用いただけます。  
携帯電話からもご利用いただけます。

■〈PHS・IP電話やファクシミリをご利用〉または〈沖縄・奄美地区の方〉は…

|          | PHS/IP電話                                      | ファックス            |
|----------|-----------------------------------------------|------------------|
| 東日本地区→   | 043 - 299 - 3863                              | 043 - 299 - 3865 |
| 西日本地区→   | 06 - 6792 - 5511                              | 06 - 6792 - 3221 |
| 沖縄・奄美地区→ | 「那覇サービスセンター」098 - 861 - 0866 (月～金 9:00～17:40) |                  |

受付時間 ●月曜～土曜:9:00～20:00 ●日曜・祝日:9:00～18:00 (年末年始を除く)

●電話番号・受付時間などについては、変更になることがあります。(2008.12)

## シャープ株式会社

本 社 〒545-8522 大阪市阿倍野区長池町22番22号  
情報システム事業本部 〒639-1186 奈良県大和郡山市美濃庄町492